UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

BIOPROCESS MONITORING IN THE BIOPHARMACEUTICAL INDUSTRY USING FILTERING AND ONLINE COST-SENSITIVE SUPERVISED LEARNING

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements

for the Degree of

MASTER OF SCIENCE

BY

PAUL OKAFOR

Norman, Oklahoma

2024

BIOPROCESS MONITORING IN THE BIOPHARMACEUTICAL INDUSTRY USING FILTERING AND ONLINE COST-SENSITIVE SUPERVISED LEARNING

A THESIS APPROVED FOR THE
GALLOGLY COLLEGE OF ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Talayeh Razzaghi, Chair

Dr. Charles Nicholson

Dr. Rui Zhu

Acknowledgements

I am deeply grateful to God for His guidance and blessings throughout this journey. I would also like to express my heartfelt thanks to my family for their unwavering love, support, and encouragement, which have been my constant source of strength.

My sincere gratitude goes to my advisor, Dr. Talayeh Razzaghi, for her invaluable guidance, mentorship, and patience, which have been instrumental in the completion of this thesis. I am also thankful to my committee members, Dr. Charles Nicholson and Dr. Rui Zhu, for their insightful feedback and support, which have greatly enriched my work.

Finally, I would like to thank my friends for their encouragement and companionship, which made this journey more enjoyable and fulfilling.

Table of Contents

List of Tables

List of Figures

Abstract

This thesis explores the enhancement of bioprocess monitoring in biopharmaceutical production using advanced filtering techniques and cost-sensitive learning algorithms. The first study addresses the challenge of optimizing recombinant protein production in *Escherichia coli* (*E. coli*) fermentations, utilizing Kalman, particle, bilateral, and extended Kalman filters to improve titer estimation accuracy. Using fermentation data from Cytovance Biologics, the proposed techniques effectively impute missing titer values and are validated on a secondary dataset from Chinese Hamster Ovary (CHO) cell lines, demonstrating their robustness and accuracy in bioprocess monitoring. In the second study, cost-sensitive learning algorithms—specifically, variants of Passive-Aggressive (PA) and Cost-Sensitive Online Gradient Descent (CSOGD)—are applied to control chart pattern recognition (CCPR) for process control. These algorithms outperform their standard non-cost-sensitive counterparts across various abnormal control chart patterns, including uptrend, downtrend, upshift, downshift, cyclic, and stratification. The inclusion of class-specific penalties and surrogate hinge loss functions enhances classification performance, particularly in smaller window lengths ($w \leq 25$). Additionally, cost-sensitive algorithms demonstrate stable performance across dynamic imbalanced datasets, though they incur higher computational costs due to increased complexity. Together, these studies provide a comprehensive framework for optimizing both the biological aspects of protein production and the operational stability of manufacturing processes, significantly advancing bioprocess monitoring in the biopharmaceutical industry.

Chapter 1: Introduction

Bioprocess monitoring has become an integral component of the biopharmaceutical industry, enabling real-time insights into complex production processes. As biopharmaceutical products are increasingly in demand, efficient monitoring systems are critical to ensuring consistent product quality and operational efficiency. Advances in bioprocess monitoring have led to the integration of sophisticated techniques that provide better control over variables such as biomass concentration, product titer, and metabolic states. This thesis focuses on two key research projects that aim to enhance bioprocess monitoring: (1) *"Monitoring Recombinant Protein Titer in Escherichia coli Fermentations Using Filtering Techniques"* and (2) *"Cost-Sensitive Online Learning for Process Monitoring."* By integrating insights from both fields, this thesis provides a unified framework for real-time monitoring of critical bioprocesses, leveraging both filtering techniques and advanced machine learning approaches for control chart pattern recognition (CCPR).

Recombinant protein production is a cornerstone of the biopharmaceutical industry, playing a critical role in the development of therapeutic agents, vaccines, and industrial enzymes. *Escherichia coli* (*E. coli*) is widely used as a production host due to its well-documented genetics, rapid growth, and cost-efficient cultivation, making it indispensable for large-scale production [15]. Approximately 30% of recombinant biopharmaceutical proteins, including key products like antigen-binding fragments, are produced using *E. coli* [16]. Its genetic simplicity and availability of mutant strains allow researchers to fine-tune metabolic pathways to increase protein yield and minimize unwanted by-products [26]. However, the process faces challenges such as inclusion body formation and metabolic imbalances, which can degrade the yield and quality of functional proteins [25]. These challenges underscore the need for advanced monitoring techniques to optimize and control recombinant protein production efficiently.

Filtering techniques offer a promising solution for enhancing bioprocess monitoring in recombinant protein production, providing real-time estimates of critical process variables that are difficult to measure directly [30]. Kalman filters, for example, have been effectively applied to fed-batch fermentations, offering accurate state

estimation for variables like biomass concentration and acetate levels in *E. coli* cultures [24, 10]. By improving the ability to monitor and control dynamic bioprocess conditions, these filtering techniques contribute to more efficient recombinant protein production, aligning with the broader goals of biopharmaceutical manufacturing to achieve high yields and product quality. However, while optimizing protein expression is essential, ensuring process stability and consistency across the entire production system is equally critical, which leads to the need for advanced process monitoring tools.

In parallel with the challenges of optimizing protein expression in *E. coli*, control charts have long been employed in the biopharmaceutical industry to monitor and maintain the quality of production processes. Developed by W.A. Shewhart in the 1920s, control charts are essential tools in Statistical Process Control (SPC), enabling real-time detection of both normal process variations and anomalies due to special factors [17]. This ability to promptly identify abnormal control chart patterns (CCPs), such as trends, shifts, and cycles, is crucial for maintaining the high-quality standards required in biopharmaceutical production [33]. As manufacturing processes become more complex, CCPR has gained attention for its role in improving decision-making and maintaining product quality.

Abnormal CCPs represent process irregularities that, if left undetected, can lead to significant inefficiencies and disruptions in production. These patterns, including trends from equipment wear or shifts due to material changes, require swift recognition to prevent costly downtime [1]. Advanced machine learning models have been increasingly applied to automate CCPR tasks, offering real-time adaptability to changing process conditions. Techniques such as Passive-Aggressive (PA) learning and Online Gradient Descent (OGD) allow for incremental updates to models as new data is received, making them well-suited for dynamic and fast-evolving production environments [8]. However, to address the inherent data imbalances often encountered in CCPR, cost-sensitive learning approaches have been developed, ensuring that anomalies are detected accurately without overwhelming the system with false positives [34].

Together, these two areas of study—optimizing recombinant protein titer using advanced filtering techniques and maintaining process stability through cost-sensitive CCPR—complement one another in enhancing bioprocess monitoring. Both projects aim to address the unique challenges of biological variability and process control, providing an integrated framework for improving the efficiency and consistency of biopharmaceutical production. The thesis is structured into two chapters: Chapter 1 discusses the use of filtering techniques in recombinant protein production in *E. coli* fermentations, while Chapter 2 focuses on applying cost-sensitive online learning for real-time control chart pattern recognition. Each chapter covers the relevant literature, data, methodology, and computational results, with a unified conclusion that integrates the findings and highlights

their contributions to improving bioprocess monitoring and operational efficiency in the biopharmaceutical industry.

Chapter 2: Monitoring Recombinant Protein Titer in Escherichia Coli Fermentations

2.1 Literature Review

Recombinant protein production is vital for pharmaceutical, agricultural, and industrial sectors. *E. coli* is the most widely used production host for this purpose due to its well-characterized genetics, rapid growth, and cost-effective cultivation [15]. About 30% of recombinant biopharmaceutical proteins, particularly antigen-binding fragments, are produced in *E. coli* [16]. Additionally, *E. coli* can express a wide range of proteins, including those that are difficult to produce in other systems. The simplicity of its genome and the availability of various mutant strains allow researchers to finely tune metabolic pathways to enhance protein yields, reduce by-products, and minimize metabolic burden [26]. However, achieving high yields of soluble, functional proteins in *E. coli* remains challenging due to inclusion body formation and metabolic imbalances that can negatively affect protein yield and quality [25].

Optimizing protein expression in *E. coli* often requires labor-intensive, trial-and-error experiments, which are time-consuming and resource-intensive. Traditional approaches focus on adjusting key parameters such as temperature, pH, induction time, and media composition [7]. However, the complex interactions between these factors necessitate more systematic and predictive approaches.

More advanced approaches to optimizing recombinant protein titer in *E. coli* fermentations rely on both empirical and physics-based models. Empirical techniques, such as Response Surface Methodology (RSM) and Design of Experiments (DoE), utilize systematic experimental designs to explore the effects of various factors

on protein production [5]. Physics-based models, including kinetic modeling and metabolic flux analysis (MFA), aim to predict optimal conditions by understanding the underlying biochemical pathways and cellular processes [15]. However, these strategies often encounter significant challenges, such as the complexity of metabolic networks, variability in strain performance, and the formation of inclusion bodies, all of which can reduce the yield of functional proteins [25]. These limitations highlight the need for more robust and adaptive methods to optimize recombinant protein titer in *E. coli* fermentations.

Advanced filtering techniques have emerged as powerful tools for state estimation in complex biotechnological processes. These techniques are particularly useful in scenarios where direct measurement of key variables, such as biomass concentration and specific growth rates, is difficult or impossible [30]. Kalman filters, for example, have been applied in various contexts, including monitoring fed-batch fermentation processes, where they aid in predicting system states with limited and noisy data [10]. Kalman filters have also been used successfully to predict and control fermentation dynamics. Kager et al. [24] have applied Kalman filters in *Penicillium chrysogenum* fed-batch cultivation for penicillin production, estimating biomass and substrate concentrations to enhance process control. Dewasme et al. [10] have utilized Extended Kalman Filters (EKFs) to estimate acetate concentrations in *E. coli* cultures by integrating biomass and gas sensor measurements, effectively addressing the lack of reliable acetate probes. EKFs have also been employed in recombinant adeno-associated virus (rAAV) production to estimate metabolite concentrations and rAAV production yields based on viable cell density in HEK293 cultures [20].

Kager et al. [23] have utilized the particle filter method for biomass, precursor, and product concentration estimation in *Penicillium chrysogenum* fed-batch cultivation, and integrated online and offline measurements for improving accuracy. Particle filters, known for their ability to handle non-linear and non-Gaussian systems [30], offer a flexible alternative, particularly in bioprocess monitoring where process dynamics can be highly unpredictable. Simutis et al. [30] have employed particle filters for biomass and specific growth rate estimation in hybridoma cell cultures, demonstrating lower estimation errors and simpler tuning procedures compared to EKFs. Despite these advances, the application of these filtering techniques in recombinant protein production—especially in *E. coli* fermentations—remains underexplored. Most studies have focused on other organisms or processes, leaving a gap in the literature regarding their efficacy in optimizing recombinant protein titer in *E. coli* [29].

This study aims to address these gaps by developing a monitoring framework that utilizes Kalman, particle, bilateral, and extended Kalman filters to estimate recombinant protein titer in *E. coli* fermentation. By applying these filters to datasets collected from the biomanufacturing industry, which contain a high percentage

of missing values, and validating them against more complete datasets, this research makes a novel contribution to the field. The remainder of this chapter is organized as follows: Section 2.2 describes the data, and Section 3.4 presents our proposed method. Section 2.4 provides the computational results, and Section 2.5 presents a discussion.

2.2 Data

The data for this study, obtained from Cytovance Biologics, includes 7 fermentation experiments to produce recombinant protein in *E. coli*. The fermentation process employs the alkaline phosphatase (phoA) [31] expression system, with experiments conducted under two sets of run conditions, labeled as project G and project S. Project G consists of four experiments using four fermenters each, adding up to 16 fermenters. Project S includes three experiments: two with six fermenters each and one demonstration (demo) with four fermenters, also adding up to 16 fermenters. The growth of the cell culture is monitored throughout the fermentation process, with samples taken at varying intervals, typically every two hours. Metabolite levels, including glucose, glycerol, phosphate, and acetate, are measured using the CEDEX Bioflow system. The titer analysis is conducted on selected samples that are first derived from SDS-PAGE gel based on the expression of the protein. The titer values are then obtained by Cation Exchange Chromatography (CEX) technique.

The purity of protein samples are estimated by visualizing the width of the protein bands when running an SDS-PAGE gel. The sample hour with the highest expression, along with those selected during the induction phase, are identified and sent for titer analysis. In the PhoA process, induction typically begins 18–24 hours into the fermentation run. Protein expression is not observed from initial hours until the induction point, which varies during the production process. Once induction occurs, protein expression starts at variable concentrations, which is influenced by several factors such as cell growth, metabolite levels, phosphate availability, feed rate, and other process parameters. Each experiment is conducted over a 48-hour period that systematically records both controlled input variables and generated outputs. The input variables include vessel type, fermenter volume, agitation speed, dissolved oxygen (DO) levels, pH, gas flow rate, and media components. The measured outputs include optical density at a wavelength of 600 nm (OD600nm), wet cell weight (WCW), protein titer, and metabolite concentration. These measured outputs are crucial for assessing fermentation process efficiency and productivity.

A spectrophotometer is used to measure the cell density while passing a visible light at a specific wavelength

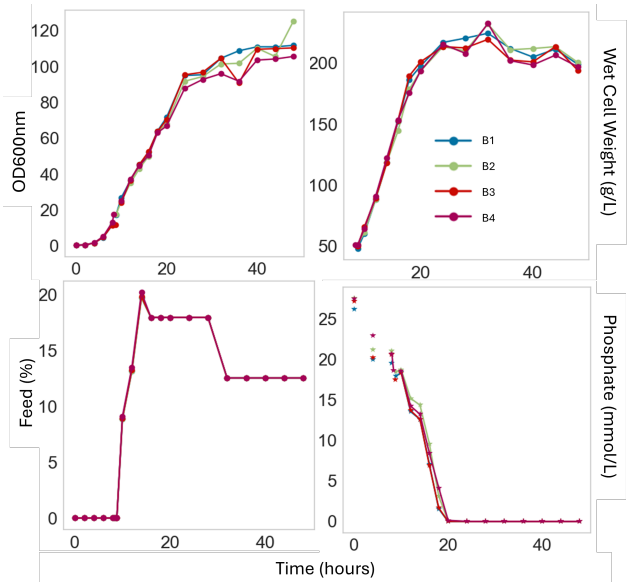


Figure 2.1: Time series plots of OD600nm, wet cell weight, feed, and phosphate variables across four experiments.

through a sample. Basically, it quantifies the amount of light that has been absorbed by the cells and estimates their concentration at a wavelength of 600nm. The WCW in protein titer production is the total mass of the cells, including their intracellular water, after they have been harvested from the fermentation or cultivation process. WCW is important in protein titer production because it can provide an indication of overall cell growth and productivity, helping to assess the health of the cells and their capacity for protein expression. The higher the WCW, the more biomass is available for potential protein production. Table 2.1 provides a summary of the data. For the rest of the paper, we will refer to this dataset as the *E. coli* data for convenience.

Table 2.1: Data characteristics of recombinant protein production in *E. coli*. Miss. represents missing data.

Variable	**Miss.**	**Mean**	**Variance**	**Median**
OD600nm	10%	53.3	1862.2	52.0
WCW	43%	170.2	3375.2	140.0
Agitation	0%	1088.6	42819.2	1185.1
Air	0%	93.4	245.9	99.6
DO	0%	53.4	511.4	41.6
Gas Flow	0%	4.4	1.6	5.0
Oxygen	0%	5.7	168.8	0.4
pH	0%	6.6	0.05	6.7
Feed	0%	10.8	76.6	12.5
Temperature	0%	29.6	1.6	30.0
Glycerol	75%	0.8	4.1	0.0
Glucose	24%	3.8	63.9	0.0
Acetate	24%	25.8	8084.4	1.9
Phosphate	29%	10.2	113.8	7.0
Titer	97%	0.6	0.6	0.4

Figure 2.1 shows the time plots of four key variables over the 48-hour fermentation period across four batches of experiments ($B1$-$B4$ from project G). OD600nm and WCW indicate continuous cell growth, which are crucial for protein production as supported by literature [16, 5]. The feed, consisting of a 50% glucose solution, was added whenever acetate levels fell below 5 mmol/L. The feed percentage rises sharply initially, then stabilizes, reflecting controlled feeding for optimal growth. The decline in phosphate concentration, particularly when levels fall below 5 mmol/L, signals the onset of protein production. Phosphate levels are monitored every four hours from the start of the experiment until depletion. At around 8 hours, after glycerol is depleted from the batch media and glucose is added, phosphate levels are monitored hourly. The phoA induction process is initiated at 20 hours when phosphate levels drop to zero. Figure 2.2 shows a strong positive correlation between OD600nm and wet cell weight (0.79), as well as negative correlations between phosphate and OD600nm (-0.85) and between phosphate and wet cell weight (-0.88), indicating that phosphate is a limiting nutrient in protein synthesis. Unlike other variables tracked bi-hourly, titer values are recorded only at the end of the 48-hour experiment period.

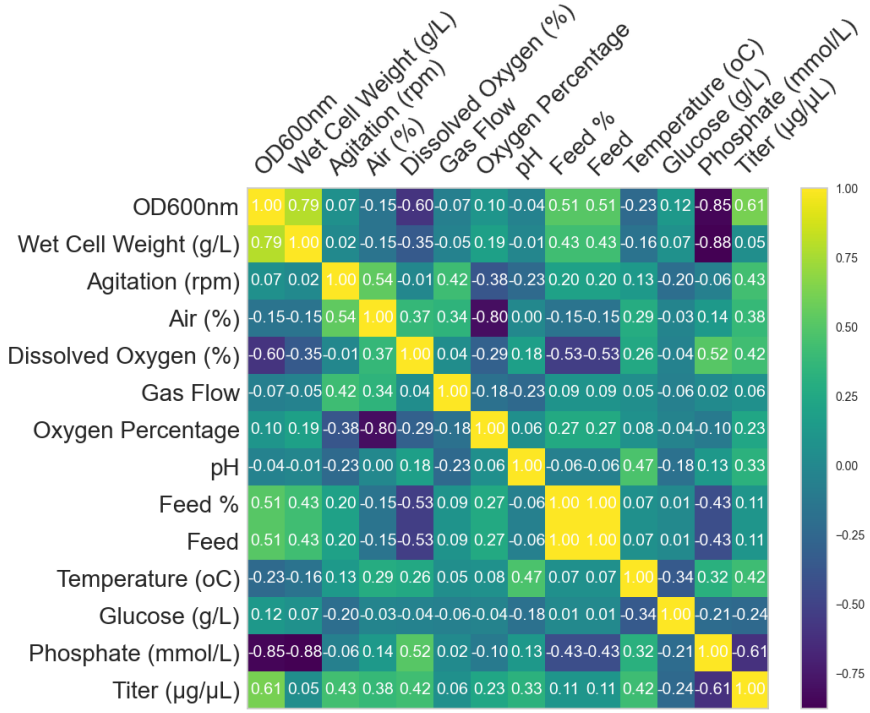


Figure 2.2: Correlation matrix in *E. coli* data

The sensitivity of titer assays, along with the need for specialized equipment and labor, further complicates frequent data collection [7, 16], hindering real-time monitoring. The missing rate in titer data is exceeds 90%. Therefore, effective imputation technique is crucial for optimizing recombinant protein titer production in *E. coli* fermentation due to substantial missing titer data.

2.2.1 CHO Dataset

Given the high rate of missing titer values in the *E. coli* dataset, a secondary dataset with more complete titer values is used to validate the proposed filtering techniques. This dataset, also provided by Cytovance Biologics, contains data from Chinese Hamster Ovary (CHO) cell lines, specifically the CHO-S strain. Compared to the *E. coli* dataset, the CHO dataset—with 50% missing titer values across 640 records from four experiments (E_1-E_4)—provides a more reliable basis for validation.

2.3 Methods

In this section, filtering techniques to impute missing protein titer values are described. Since titer values are only recorded at the end of the 48-hour experiment, several assumptions are made to enable effective

imputation. Initially, titer values are assumed to be zero until significant phosphate depletion ($\leq$ 5 mg/ml) signals the start of protein synthesis. End-point titer values are estimated within a range of 0.3 to 2.2 $\mu g/\mu L$, based on empirical data and expert knowledge, ensuring realistic bounds. These assumptions form the basis for applying the filtering techniques for imputation.

2.3.1 Kalman Filters

Kalman filters (KFs), known for efficient real-time state estimation [30], are employed to impute missing protein titer data in *E. coli* [28, 20]. The standard Kalman filter algorithm identifies optimal estimates of both measured and unmeasured state variables by integrating data from a linear stochastic difference model with real-time measurements [21]. The state vector $\mathbf{x}_t$ is the protein titer at time t. The system model describes the evolution of the state from time t to time $t+1$, formulated as

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t \tag{2.1}$$

where A is the state transition matrix, B is the control input matrix, $\mathbf{u}_t$ represents control inputs (e.g., pH, temperature, dissolved oxygen) at time t, and $\mathbf{w}_t$ is the process noise at time t with covariance Q. The system model is combined with the measurement model, which explains the relationship between the state and the measurement at time t as follows

$$\mathbf{z}_t = H\mathbf{x}_t + \mathbf{v}_t \tag{2.2}$$

where $\mathbf{z}_t$ is the measurement variable at time t, H is the observation matrix, and $\mathbf{v}_t$ is the measurement noise at time t with covariance R. The KF algorithm consists of two main steps: [37]: (1) The system model predicts the state variables and the estimation error covariance until the initial measurement is obtained (prediction step), (2) Predicted model estimates are integrated with measured values to produce corrected estimates (correction step).

2.3.2 Particle Filters

Particle filters (PFs), also known as Sequential Monte Carlo methods, aestimate the state of non-linear and non-Gaussian systems by approximating the posterior distribution of state variables using a set of weighted particles [11, 30]. These methods have been utilized to impute missing protein titer values during *E. coli* fermentation [6]. The PF algorithm involves the following steps [30]: (1) **Initialization:** Generate an initial

population of state vectors (particles) $\{x_0^i\}_{i=1}^N$, and set the initial weight of the i-th particle as $w_0^i = \frac{1}{N}$. We assume $x_0^i \sim \mathcal{N}(0, Q)$.

(2) **Prediction:** predict state of the i-th particle (x_t^i) at time t through the process model $x_t^i = f(x_{t-1}^i, u_{t-1}) + w_{t-1}^i$, where $w_{t-1}^i \sim \mathcal{N}(0, Q)$.

(3) **Weight Update:** Update the weight of the i-th particle at time t based on the likelihood of observations $_t^i = w_{t-1}^i \cdot p(y_t \mid x_t^i)$, where $p(y_t \mid x_t^i)$ is the likelihood of the observation y_t given the predicted state x_t^i. Normalize the weights so that $\sum_{i=1}^N w_t^i = 1$.

(4) **Resampling:** Resample the particles based on their weights to avoid particle degeneracy [4].

(5) **Estimation:** The state estimate ($\hat{x}_t$) at time t is given by the weighted mean of the particles $\hat{x}_t = \sum_{i=1}^N w_t^i x_t^i$. The likelihood function is modeled as $p(y_t \mid x_t^i) \sim \mathcal{N}(h(x_t^i), R)$, where y_t is the measurement at time t and R is the measurement noise covariance. $h(x_t^i)$ is the measurement function mapping the state x_t^i to the observation space [30]. This method is ideal for estimating missing titer data in non-linear fermentation processes [30].

2.3.3 Bilateral Filters

Bilateral filters (BFs), a type of spatial filter, are well-known for their edge-preserving properties, making them perfect for noise reduction and signal smoothing without blurring important edges [40, 27]. In the context of optimizing recombinant protein titer in *E. coli* fermentation, BFs can effectively impute missing protein titer values while preserving the integrity of key features in the data [27]. The filtered output $\mathbf{y}_i$ for a data point i is calculated as a weighted average of neighboring data points, with the weights are determined by spatial distance and intensity difference [32], as follows

$$\mathbf{y}_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_s^2}\right) \exp\left(-\frac{|I_i - I_j|^2}{2\sigma_i^2}\right) \mathbf{x}_j \tag{2.3}$$

where W_i is the normalization factor defined as

$$W_i = \sum_{j \in \mathcal{N}(i)} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_s^2}\right) \exp\left(-\frac{|I_i - I_j|^2}{2\sigma_i^2}\right) \tag{2.4}$$

Here, $\mathbf{x}_i$ and $\mathbf{x}_j$ are the input values at positions i and j, I_i and I_j are the intensity values, σ_s is the spatial standard deviation, and σ_i is the intensity standard deviation [27]. The hyperparameters σ_s and σ_i are crucial

for the filter's effectiveness and are determined through grid search methods [14, 12].

2.3.4 Extended Kalman Filters

Extended Kalman Filters (EKFs) extend the standard Kalman filter to handle non-linear systems by approximating non-linear state transitions and observation models using their linear counterparts via Jacobian matrices [22]. EKFs are particularly useful for real-time estimation in systems where underlying processes are non-linear, which is common in biological processes [18]. In EKFs, the system model evolves the state vector $\mathbf{x}_t$ through a non-linear state transition function f, given by

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{w}_t \tag{2.5}$$

where $\mathbf{u}_t$ represents the control input, and $\mathbf{w}_t$ is the process noise with covariance Q. The observation model (see Eq. 2.6), which relates the state to the measurements, is also non-linear.

$$\mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{v}_t \tag{2.6}$$

where h is the non-linear measurement function, and $\mathbf{v}_t$ is the measurement noise with covariance R. The EKF is based on linearizing the non-linear functions f and h at each time step using their respective Jacobian matrices. These linear approximations enable the EKF to estimate the state in non-linear systems with computational efficiency comparable to that of the standard Kalman filter.

2.4 Computational Results

The filtering techniques to estimate recombinant protein titer ($\mu g/\mu L$) in *E. coli* fermentation are implemented in Python using the NumPy library. All experiments are performed on a laptop with an AMD Ryzen 5 7535HS processor, 64 GB of RAM, and a 64-bit operating system. The optimal hyperparameters for each filtering technique are determined through grid search. For KF, control inputs are pH, temperature, and dissolved oxygen percentage. The state transition and observation matrices are set as identity matrices, with the process noise covariance $Q = 0.01I$ and the measurement noise covariance $R = 0.01I$, where I represents the identity matrix. In PF, the optimal number of particles is set to 1,000, with process noise and measurement noise standard deviations of 0.1 and 0.01, respectively. In BF, a window size of five and standard deviations of one are employed for both spatial and intensity parameters to balance smoothing and detail preservation.

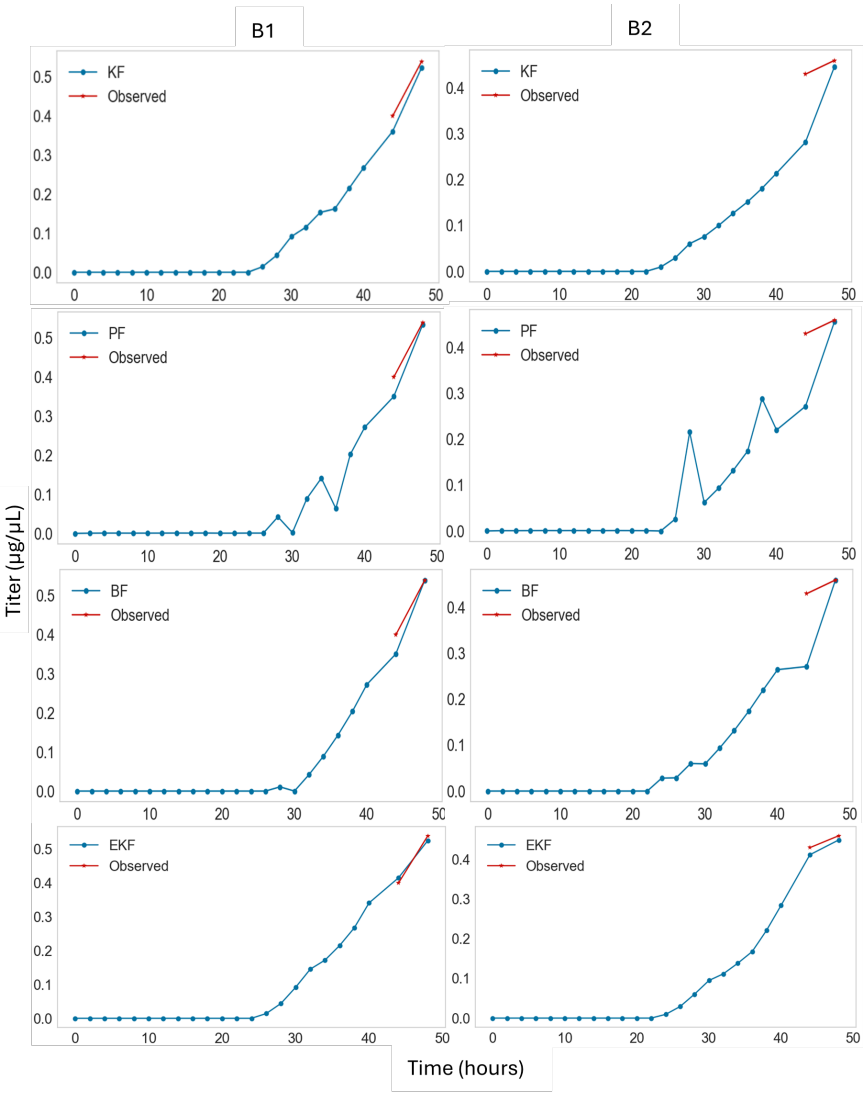


Figure 2.3: Estimated titer values for two experimental batches (B_1 and B_2) in *E. coli* data using Kalman Filter (first row), Particle Filter (second row), Bilateral Filter (third row), and Extended Kalman Filter (fourth row). The red line represents the true observed values.

Figure2.3 illustrates the estimated titer values of two experimental batches (B_1 and B_2) in *E. coli* data using KF, PF, BF, and EFK. The EKF provides a closer fit to the observed data compared to other methods, while the results of the KF, BF, and PF are relatively similar. Moreover, the EKF shows a smoother growth in titer values over time, with fewer fluctuations compared to the other filtering techniques.

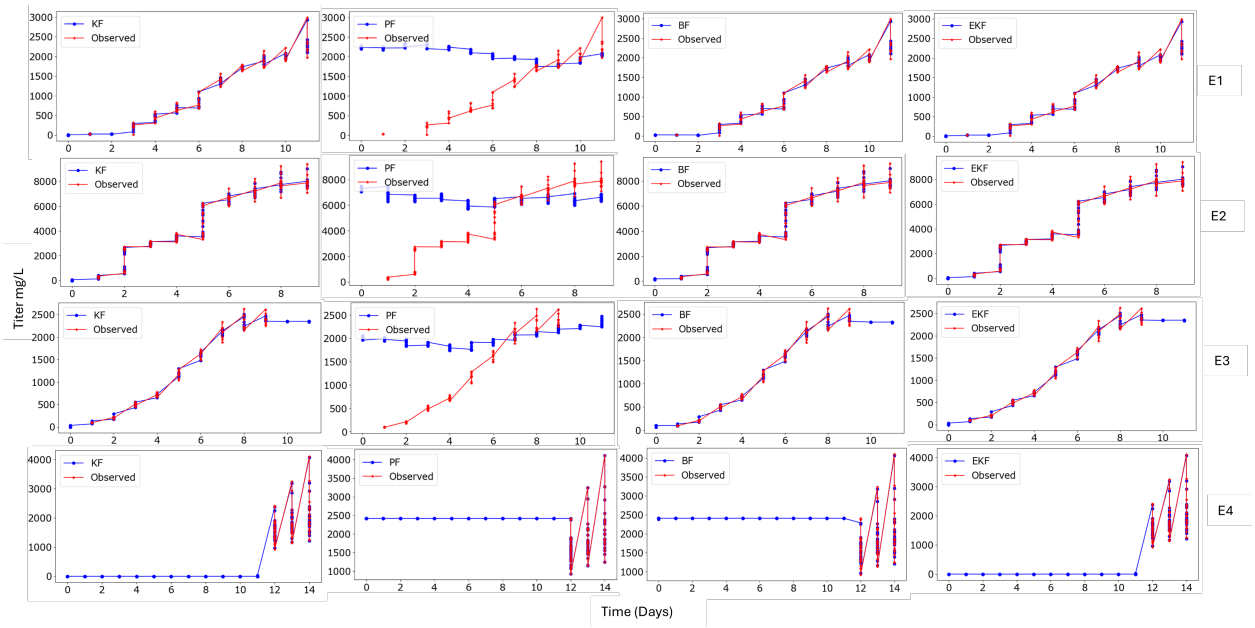


Figure 2.4: Estimated titer values for four experiments E_1-E_4. Left column shows the KF, middle columns show the PF and BF respectively, and the right column shows the EKF results. The red line represents the true observations.

2.4.1 Validation Using CHO Data

Additionally, we evaluated the filtering techniques using the CHO dataset, which had fewer missing titer values. Figure 2.4 presents the estimated titer values for four experiments (E_1 - E_4) using KF, BF, PF, and EKF. The figure is organized into four columns: the left column displays the KF results compared to the observed titer values, the middle columns show the results of PF and BF, respectively, and the right column presents the EKF results.

According to the figure, all filtering methods demonstrate a close fit to the observed data, except for PF in experiments E_1 - E_3, where PF shows a notably inferior fit. This is consistent with the results in Table 2.2, which indicates that BF performs slightly better in terms of RMSE, MAE, and MAPE, though the differences are not significant when compared to KF and EKF. While PF results were inferior to the other techniques in terms of RMSE and MAE, it outperformed them in terms of MAPE.

2.5 Discussion

This study explores the application of advanced filtering techniques to optimize recombinant protein titer in *E. coli* fermentation. Specifically, Kalman filters, particle filters, and bilateral filters are proposed to impute missing titer values, addressing the high proportion of missing data frequently encountered in biotechnological

Table 2.2: RMSE, MAE, and MAPE results for KF, BF, PF, and EKF for the CHO dataset for four experiments. The best results are highlighted in bold font.

Experiment	Filter	RMSE	MAE	MAPE
E1	KF	79.47	64.43	14.75%
	PF	1273.54	992.58	**12.07%**
	BF	**79.42**	**64.27**	13.92%
	EKF	79.47	64.43	14.75%
E2	KF	203.40	159.01	5.07%
	PF	3234.39	2520.99	**3.55%**
	BF	**203.29**	**158.45**	4.83%
	EKF	203.40	159.01	5.07%
E3	KF	76.07	57.41	6.66%
	PF	1063.46	841.14	**3.66%**
	BF	**75.72**	**56.72**	6.20%
	EKF	76.07	57.41	6.66%
E4	KF	104.42	84.21	5.08%
	PF	967.41	857.34	**0.55%**
	BF	**102.47**	**82.56**	4.95%
	EKF	104.42	84.21	5.08%

processes. In the *E. coli* dataset, the EKF provided robust titer value estimates while effectively preserving growing trends. This observation is consistent with expert opinion on state estimation and control in bioprocessing. Additionally, the validation confirms that the assumptions made for the *E. coli* dataset did not compromise the accuracy of the imputed values, further demonstrating the reliability of these advanced filtering methods.

In the CHO dataset, BF shows a slight improvement in performance metrics compared to the KF and EKF methods, though the differences are not statistically significant. Notably, BF estimates effectively reduce noise while preserving key data features. The application of BFs for noise reduction and detail preservation is well-studied in fields such as medical imaging and computer vision [32]. This research extends the application of BFs to bioprocess optimization, demonstrating their effectiveness in preserving data variability within biotechnological processes. Furthermore, validation using the CHO dataset, which had fewer missing values, confirmed the robustness and accuracy of these filtering techniques. The filters produced results that closely aligned with the observed titer values

Previous studies have demonstrated that the application of advanced filters in bioprocess optimization enhances the stability and efficiency of production processes by providing accurate and timely state estimations [2, 30]. KFs, in particular, have shown significant improvements in process control by providing precise estimations of fermentation dynamics. PFs are known for their adaptability to dynamic changes in bioprocesses, making them well-suited for optimizing protein yield in fermentation settings [30, 6, 32].

Chapter 3: Cost-Sensitive Online Learning for Process Monitoring

3.1 Literature Review

Control charts are fundamental tools in statistical process control (SPC), which are used to maintain the stability of production processes by monitoring key quality characteristics. Developed in the 1920s by W.A. Shewhart, control charts serve to distinguish between normal variations in a process and those caused by specific malfunctions [17]. These charts show the behavior of critical process variables over time, and when the process moves out of control, the control chart pattern (CCP) can indicate the type of anomaly occurring in the system [33]. Among the most widely recognized patterns are normal, cyclic, upward trend, downward trend, systematic, stratification, upward shift, and downward shift patterns [1]. The early detection and identification of these patterns can help to prevent significant disruptions in production, leading to improved operational efficiency [39]. As processes become more complex, CCPR has gained attention for its role in improving decision-making and maintaining product quality.

Abnormal CCPs represent process irregularities that require immediate attention to prevent costly failures. Trend patterns, such as upward and downward trends, typically result from gradual wear and tear on equipment or tool degradation, while shift patterns indicate abrupt changes in the process, such as variations in material or sudden machine malfunctions [1]. Cyclic patterns often suggest periodic fluctuations due to environmental factors or rotating operators, whereas systematic patterns reveal regular fluctuations, which might occur due to consistent errors in measurement tools or process settings [33, 13]. Correctly identifying these patterns is critical in modern production environments where abnormal patterns may arise from multiple sources simultaneously [39]. Therefore, advanced CCPR techniques, often employing machine learning models, have been developed to efficiently recognize these patterns and minimize the likelihood of

production downtime [9].

As control chart monitoring systems require real-time updates, traditional batch learning approaches often fall short when dealing with rapidly changing processes. To address this, online learning has emerged as an efficient solution for pattern recognition in dynamic environments. Unlike batch learning, where models are trained on historical data in offline mode, online learning continuously updates its models based on new incoming data [8]. This makes it particularly suitable for CCPR, where patterns may evolve over time, and quick adaptation is crucial. Online classification algorithms, such as Passive-Aggressive (PA) learning and Online Gradient Descent (OGD), have shown effectiveness in real-time CCPR tasks. The PA algorithm, in particular, is designed to handle noisy data and perform updates incrementally, making it robust for tasks where the data distribution may shift over time [8]. While these models are successful, they may struggle in highly imbalanced datasets, this challenge is commonly addressed through cost-sensitive learning approaches [34].

In industrial applications, the misclassification of certain abnormal patterns can have varying levels of consequences. Cost-sensitive learning addresses this by factoring in the costs associated with different types of misclassification errors. In the context of CCPR, a false positive for a minor deviation may be less costly than missing a critical process failure. Cost-sensitive online learning builds upon the online learning framework but adjusts for the cost of errors by minimizing weighted misclassification costs [34]. This approach is essential in settings where the failure to detect critical patterns, such as sudden shifts or trends, can result in significant operational downtime or damage. By assigning different costs to errors based on the importance of the patterns, algorithms like Cost-Sensitive Online Gradient Descent optimize both efficiency and risk management [34]. These advancements allow industries to prioritize more severe patterns and minimize economic losses, ensuring timely corrective actions in high-stakes environments.

CCPR has a wide range of real-world industrial applications, especially in manufacturing, healthcare, energy, and biopharmaceuticals. In manufacturing, CCPR helps in identifying equipment malfunctions, material defects, or variations in production quality early, allowing companies to take corrective measures before costly failures occur. Similarly, in healthcare, the use of CCPR in patient monitoring ensures early detection of adverse health events, improving patient outcomes by enabling timely interventions. In the biopharmaceutical sector, cost-sensitive online learning proves invaluable, particularly in monitoring recombinant protein manufacturing processes. Here, control charts track variables such as protein titer, pH levels, and temperature to ensure consistent product quality. Given the high stakes in this industry—where deviations in these variables can lead to significant financial losses—cost-sensitive methods prioritize critical anomalies while

reducing false positives for less impactful deviations. This focus on minimizing economic losses and enhancing process reliability is crucial in high-stakes environments like biomanufacturing, where even minor errors can disrupt production and result in costly delays.

Previous studies, such as those by Xanthopoulos and Razzaghi [38], effectively applied weighted support vector machines for CCPR but did not incorporate an online learning framework, limiting their adaptability in dynamic environments. Similarly, extensive work on online learning algorithms, such as the LIBOL library by Hoi et al. [36], and Online Passive-Aggressive Algorithms by Crammer et al. [8] covered methods like PA and OGD, yet failed to extend these to their cost-sensitive variants (especially for binary classification) [36, 19]. Although Wang et al. [34] proposed the Cost-Sensitive Online Gradient Descent (CSOGD) algorithm, it was not applied to CCPR, which is a notable gap in the literature. Moreover, while the PA algorithm shares similarities with support vector approaches, which have shown promise in CCPR, it has not been explicitly developed or widely applied in a cost-sensitive context. The novelty of this study lies in bridging these gaps by extending the cost-sensitive framework to both CSOGD and cost-sensitive passive-aggressive algorithms for CCPR, thus providing a more comprehensive and adaptive approach to pattern recognition in real-time industrial processes. This advancement addresses the limitations of prior studies, offering a more effective solution for optimizing error costs and enhancing the performance of CCPR in high-stakes environments.

This research establishes a foundation for a detailed exploration of cost-sensitive online learning in CCPR. The subsequent chapters will examine the data sources, methods used to extend existing algorithms to a cost-sensitive context, and the computational techniques applied to assess their effectiveness. The methodology will detail the design of experiments, including algorithm implementation and evaluation metrics, followed by an in-depth analysis of the results and their practical implications for real-world industrial processes.

3.2 Data

The study utilizes synthetic datasets to train and evaluate cost-sensitive online learning algorithms for CCPR. These synthetic datasets simulate various control chart patterns, providing a controlled environment for algorithm development and testing. They represent typical industrial scenarios, enabling the algorithms to learn and adapt effectively to detect and classify abnormal patterns. This approach ensures the algorithms are well-suited for CCPR tasks, offering insights into their performance across different pattern types and conditions, with a focus on maintaining process stability and monitoring critical variables.

3.2.1 Synthetic Data

The synthetic dataset consists of time series data that simulate seven commonly observed abnormal CCPs. These patterns—Uptrend, Downtrend, Upshift, Downshift, Systematic, Cyclic, and Stratification—are frequently encountered in control chart analysis across industrial applications. Each pattern is generated using predefined equations that model process behaviors under abnormal conditions. The specific equations for generating these patterns are detailed in Table 3.3. A total of 380 datasets were generated for each pattern, with each dataset corresponding to a specific combination of window length (i) and abnormal parameter (g, s, a, d, σ'). The window lengths varied from 10 to 100 in steps of 5. Each dataset consisted of 900 normal samples and 100 abnormal samples, resulting in an imbalanced classification problem that necessitated the use of cost-sensitive learning techniques.

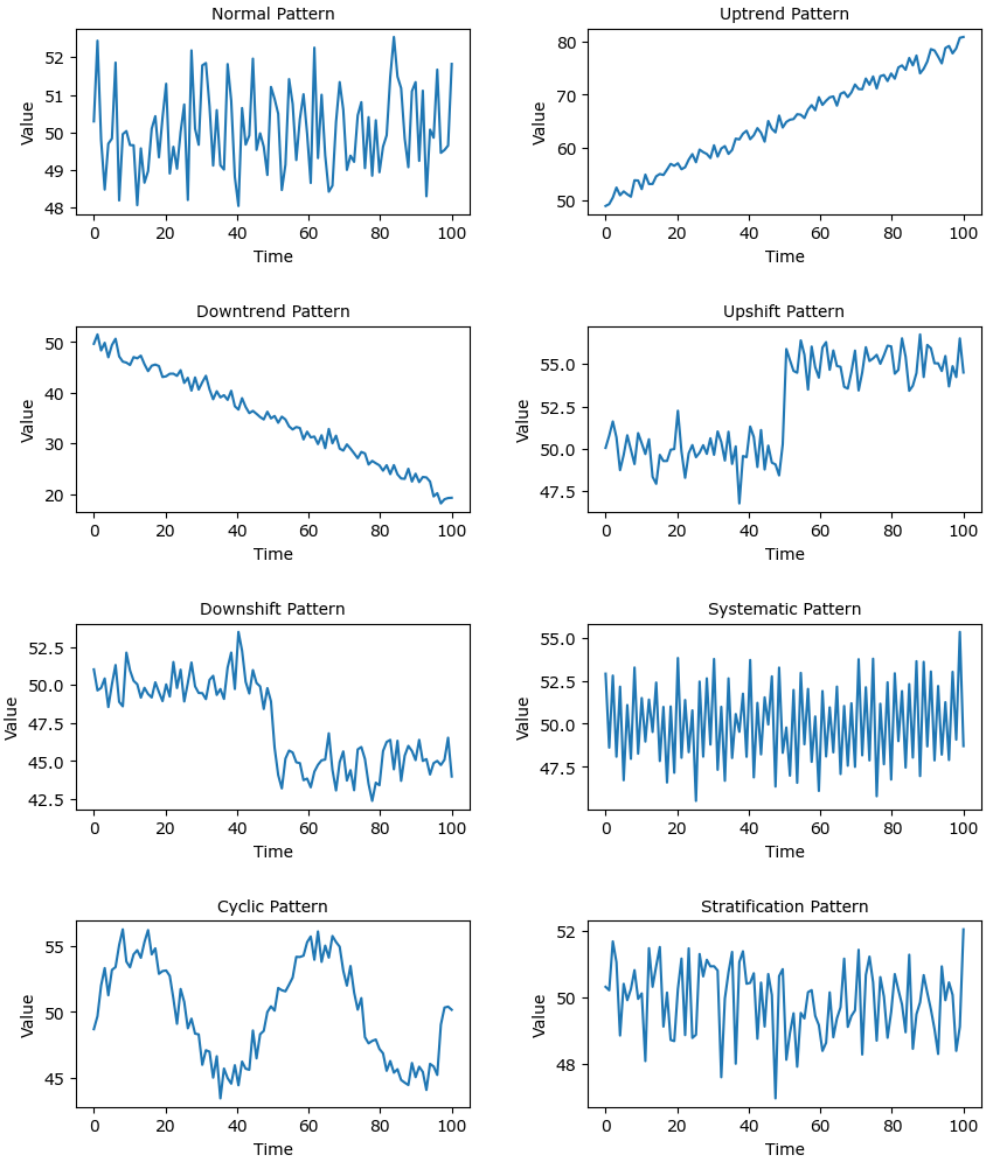


Figure 3.5: Examples of normal and abnormal control chart patterns.

Table 3.3: Mathematical models of control chart patterns [3]

Pattern	**Mathematical Model**
Normal	$y_t = \mu + r_t\sigma$
Uptrend	$y_t = \mu + r_t\sigma + gt$
Downtrend	$y_t = \mu + r_t\sigma - gt$
Upshift	$y_t = \mu + r_t\sigma + ks$
Downshift	$y_t = \mu + r_t\sigma - ks$
Systematic	$y_t = \mu + r_t\sigma + d(-1)^t$
Cyclic	$y_t = \mu + r_t\sigma + a\sin\left(\frac{2\pi t}{T}\right)$
Stratification	$y_t = \mu + r_t\sigma'$

y_t: Value of the time series for sample t with window length w;
μ: Mean of the normal process; w: Window length; r_t: Standard normal variate at window length w;
σ: Standard deviation of the normal process;
σ': Random noise for stratification with standard deviation σ', $0.1 \leq \sigma' \leq 0.5\sigma$;
g: Magnitude of gradient for trend patterns; d: Magnitude of the systematic pattern;
k: Shift parameter; s: Shift magnitude for upshift/downshift patterns;
a: Amplitude of the cyclic variation; T: Period of a cycle;

Table 3.4: Summary of parameter ranges for the control chart patterns.

Name	Symbol	Range
Window length	w	$w \in \{10, 15, \ldots, 100\}$
Process mean	μ	$\mu = 0$
Period of a cycle	T	$T \in \{8, 16\}$
Random noise (normal/systematic patterns)	r_t	$r_t \sim \mathcal{N}(0, 1)$
Random noise for stratification	r'_t	$r'_t \sim \mathcal{N}(0, \sigma')$
Magnitude of gradient for the trend (up/down trend pattern)	g	$g \in [0.05\sigma, 0.1\sigma]$
Parameter determining the shift position	k	$k \sim w/2$
Magnitude of the shift (up/down shift pattern)	s	$s \in [1.5\sigma, 3.0\sigma]$
Magnitude of the systematic pattern	d	$d \in [0.5\sigma, 3.0\sigma]$
Amplitude (cyclic pattern)	a	$a \in [0.5\sigma, 3.0\sigma]$
Stratification standard deviation	σ'	$\sigma' \in [0.1\sigma, 0.5\sigma]$

To ensure consistency in data representation, all instances in the input sequence are normalized using the Euclidean norm. This process, illustrated in Equation 3.7, ensures that each instance in the input sequence has a unit norm, which facilitates fair comparison across different patterns [8]. After normalization, the Euclidean norm of each vector equals one, as shown in Equation 3.8:

$$\|\mathbf{x}_t\| = \sqrt{\sum_{w=1}^{w} (x_{t,w})^2} \tag{3.7}$$

$$\|\mathbf{x}_t\|^2 = 1 \tag{3.8}$$

where: $\mathbf{x}_t$ is a vector associated with the sample at time t, and it has components $\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, \ldots, \mathbf{x}_{t,w}$, where w is the window length. The summation runs over all the components of the vector, from $w = 1$ to w (i.e., over the length of the window). $\mathbf{x}_{t,w}$ represents the w-th element (or component) of the vector $\mathbf{x}_t$ for the time instance t. This normalization process ensures that comparisons between different abnormal patterns are both consistent and unbiased. Each instance in the input sequence is labeled for classification purposes, where normal patterns are assigned a label of -1, and abnormal patterns—comprising any of the seven types (Uptrend, Downtrend, Upshift, Downshift, Systematic, Cyclic, or Stratification)—are labeled as $+1$. This labeling scheme enables the algorithm to effectively distinguish between normal and abnormal sequences.

3.3 Methods

The methodology involves adapting both Online Gradient Descent (OGD) and Passive-Aggressive (PA) algorithms to operate within a cost-sensitive framework for CCPR. These algorithms are modified by incorporating class-specific penalty mechanisms, which allow them to prioritize misclassification errors based on their associated costs. This adaptation enhances the detection of critical abnormal patterns in CCPs, a key requirement in streaming data environments where real-time, accurate updates to the model are essential [8, 38]. Synthetic data is generated to represent both normal and abnormal CCPs, allowing for comprehensive testing of the cost-sensitive algorithms. The abnormal patterns are tailored to reflect various real-world scenarios of CCP behavior, including trends, shifts, systematic variations, and cyclic components. Each pattern is associated with specific misclassification costs, requiring the algorithms to adjust their decision boundaries dynamically in response to evolving data streams. The goal is to optimize classification performance by minimizing the cumulative cost-sensitive loss, balancing between minimizing false positives and false negatives based on their respective costs.

The PA and OGD algorithms are modified into their cost-sensitive versions, namely Cost-Sensitive Passive-Aggressive (CSPA) and Cost-Sensitive Online Gradient Descent (CSOGD), with adjustments to their loss functions to handle imbalances in misclassification costs. This methodology involves the formulation of these algorithms, the use of synthetic CCP data, and an evaluation procedure that examines the cost-efficiency and accuracy of the framework in real-time pattern recognition tasks.

3.3.1 Cost-Sensitive Online Gradient Descent (CSOGD) Algorithms

CSOGD is an extension of the standard Online Gradient Descent (OGD) algorithm, tailored to handle cost imbalances in classification tasks. The algorithm, developed by Wang et al.[34] iteratively updates the model based on incoming instance-label pairs $(\mathbf{x}_t, y_t)$, where $\mathbf{x}_t \in \mathbb{R}^n$ represents the input features and $y_t \in \{-1, +1\}$ is the binary label. Unlike standard OGD, CSOGD incorporates class-specific penalties to adjust for misclassification costs, ensuring that errors in minority class is minimized more aggressively. The weight vector $\mathbf{w}_t$ is updated at each iteration by computing the gradient of a cost-weighted loss function, such as the hinge loss, which is commonly used for binary classification tasks [34, 8]. The update rule is expressed as:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \nabla \ell_t(\mathbf{w}_t), \tag{3.9}$$

where λ denotes the learning rate, and $\ell_t(\mathbf{w}_t)$ is the loss incurred at iteration t. For binary classification, the hinge loss is utilized and is defined as:

$$\ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = \max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)). \tag{3.10}$$

The hinge loss penalizes the model based on the margin between the true label y_t and the model's prediction $\mathbf{w}_t \cdot \mathbf{x}_t$. If the margin is less than 1, the loss is positive, and the weight vector is updated in the opposite direction of the gradient, thereby minimizing classification errors over time.

To address the imbalance in data distribution and unequal costs associated with different types of classification errors, Wang et al. [34] extended OGD to develop CSOGD algorithms. The key idea behind CSOGD is to introduce a cost-sensitive parameter ρ that adjusts the importance of different types of classification errors. Specifically, two formulations of cost sensitivity are considered. The first formulation aims to maximize the weighted sum of sensitivity and specificity, while the second formulation seeks to minimize the weighted misclassification cost. This dual objective is captured in the following equation:

$$\sum_{y_t=+1} \rho \cdot \mathbb{I}_{(y_t \mathbf{w} \cdot \mathbf{x}_t < 0)} + \sum_{y_t=-1} \mathbb{I}_{(y_t \mathbf{w} \cdot \mathbf{x}_t < 0)}, \tag{3.11}$$

where ρ is defined as:

$$\rho = \frac{\eta_p T_n}{\eta_n T_p} \quad \text{for the maximization of the weighted sum,}$$

$$\rho = \frac{c_p}{c_n} \quad \text{for the minimization of the weighted misclassification cost.}$$

Here, η_p and η_n are weights assigned to sensitivity and specificity, respectively, and T_p and T_n denote the number of positive and negative examples in the dataset. Alternatively, in the context of misclassification cost, c_p and c_n are the costs associated with false negatives and false positives, respectively. By incorporating ρ into the OGD framework, the algorithm can effectively balance the trade-offs between different types of errors, making the updates cost-sensitive [34]. To incorporate these cost-sensitive formulations into the learning process, Wang et al. [34] modified the standard hinge loss function by introducing two variants: Hinge Loss I and Hinge Loss II. The indicator function in the objective function is not convex, making optimization challenging. Therefore, the indicator function is replaced by its convex surrogate in both hinge loss variants

to solve the optimization problem effectively.

The first variant, Hinge Loss I (L1), adjusts the margin term based on the cost-sensitive parameter ρ:

$$\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t)) = \max(0, (\rho \cdot \mathbb{I}_{y_t=1} + \mathbb{I}_{y_t=-1}) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)), \tag{3.12}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. This adjustment increases the penalty for costly errors by altering the required margin for predictions based on class labels, ensuring that errors from the more important class incur greater loss.

The second variant, Hinge Loss II (L2), scales the entire hinge loss function by the cost-sensitive parameter ρ. This approach modifies the slope of the hinge loss, amplifying updates for errors with higher associated costs, thus providing a more aggressive adjustment to the learning objective:

$$\ell_{II}(\mathbf{w}; (\mathbf{x}_t, y_t)) = (\rho \cdot \mathbb{I}_{y_t=1} + \mathbb{I}_{y_t=-1}) \cdot \max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)). \tag{3.13}$$

In CSOGD, the weight update rule varies depending on the hinge loss variant. For Hinge Loss I, the margin term is adjusted to reflect ρ_t. For Hinge Loss II, the gradient is scaled entirely by ρ_t, ensuring that the cost-sensitive adjustments directly influence the update process based on the misclassification costs associated with each class. Both Hinge Loss I and II share the following weight update formula:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda \rho_t y_t \mathbf{x}_t \quad \text{if } \ell_t(\mathbf{w}_t) > 0, \tag{3.14}$$

$$\rho_t = \rho \cdot \mathbb{I}_{y_t=1} + \mathbb{I}_{y_t=-1}. \tag{3.15}$$

where ρ_t is a cost-sensitive parameter:

The CSOGD algorithms are efficient with a time complexity of $O(T \times n)$, where T is the number of instances, and n is the dimensionality of the data. The design of CSOGD ensures that the model effectively learns by accounting for the varying costs of different types of classification errors, thus optimizing the performance in cost-sensitive classification scenarios.

Algorithm 1 Cost-Sensitive OGD (CSOGD) Algorithms

Input: Learning rate λ, Bias parameter ρ:

$$\rho = \begin{cases} \frac{\eta_p T_n}{\eta_n T_p} & \text{for maximizing the weighted sum,} \\ \frac{c_p}{c_n} & \text{for minimizing the weighted misclassification cost} \end{cases}$$

Data stream $\{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$, where $\mathbf{x}_t \in \mathbb{R}^n$ and $y_t \in \{-1, +1\}$

Output: Final weight vector $\mathbf{w}_{T+1}$

Initialize weight vector $\mathbf{w}_1 = \mathbf{0}$
for $t = 1, \ldots, T$ **do** ▷ Iterate over each data point
  Receive instance $\mathbf{x}_t$ and label y_t
  Predict: $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
  Compute loss: $\ell_t(\mathbf{w}_t) = \ell^*(\mathbf{w}_t; (\mathbf{x}_t, y_t))$; $* \in \{I, II\}$ ▷ *Standard OGD* with Hinge loss if $*$ is absent
  if $\ell_t(\mathbf{w}_t) > 0$ **then** ▷ Update only if loss is positive
    Update weight vector:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \nabla \ell_t(\mathbf{w}_t)$$

  else
    $\mathbf{w}_{t+1} = \mathbf{w}_t$ ▷ No update if loss is zero
  end if
end for
return $\mathbf{w}_{T+1}$

3.3.2 Cost-sensitive Passive-Aggressive (CSPA) Algorithms

Passive-Aggressive (PA) algorithms are a family of online learning algorithms that adjust model parameters incrementally as new data arrives. Unlike traditional Perceptron methods [34, 35], PA models ensure that correctly classified instances maintain a margin from the decision boundary, which improves the robustness of the classifier in dynamic environments. The PA algorithm updates its weight vector $\mathbf{w}_t$ by solving a constrained optimization problem at each time step t, adjusting the model only when a misclassification occurs or the margin constraint is violated.

The update rule is derived from the following optimization problem:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 \quad \text{subject to} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0, \tag{3.16}$$

where $\mathbf{x}_t$ is the input instance and y_t is its corresponding label. The objective is to find the new weight vector $\mathbf{w}_{t+1}$ that is as close as possible to the current weight vector $\mathbf{w}_t$, while enforcing a classification margin. The closed-form solution for the weight update is given by:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t, \tag{3.17}$$

where the step size τ_t is computed as:

$$\tau_t = \frac{\max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t))}{\|\mathbf{x}_t\|^2} = \frac{\ell(\mathbf{w}; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2}. \tag{3.18}$$

The PA algorithm has two additional variants, PA-I and PA-II, which introduce regularization to account for noisy data and ensure stability. In PA-I, a slack variable ξ is introduced. The optimization problem and corresponding update rule becomes:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi \quad \text{subject to} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi, \quad \xi \geq 0, \tag{3.19}$$

$$\tau_t = \min\left(C, \frac{\max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t))}{\|\mathbf{x}_t\|^2}\right) = \min\left(C, \frac{\ell(\mathbf{w}; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2}\right). \tag{3.20}$$

The introduction of the parameter C prevents overly large updates, making the algorithm more robust to noisy data. It regulates the balance between aggressive updates and stability (in both PA-I and PA-II), where larger values of C lead to more aggressive corrections, while smaller values produce conservative adjustments, thereby improving the model's resistance to noise. In PA-II, the slack variable is penalized quadratically, resulting in the following update rule:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 \quad \text{subject to} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi, \tag{3.21}$$

$$\tau_t = \frac{\max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t))}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}} = \frac{\ell(\mathbf{w}; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}}. \tag{3.22}$$

To address imbalanced datasets, the CSPA algorithms modify the PA variants by incorporating class-specific weights C^+ and C^-, ensuring that instances from the minority class are given higher priority [38]. The weight update rules for CSPA-I and CSPA-II are adjusted accordingly, with class-specific penalties applied based on the instance's label:

$$C^+ = \frac{C}{n^+}, \quad C^- = \frac{C}{n^-}, \tag{3.23}$$

$$C_t = \begin{cases} C^+ & \text{if } y_t = +1, \\ C^- & \text{if } y_t = -1. \end{cases} \tag{3.24}$$

where n^+ and n^- represent the sizes of the positive (abnormal) and negative (normal) classes (analogous to η_p and η_n in the CSOGD context), and C is the base regularization parameter. This modification ensures that the algorithm assigns higher emphasis on minimizing costly misclassification errors, thereby enhancing performance in scenarios with imbalanced datasets.

For the CSPA-I and CSPA-II algorithms, the original PA-I and PA-II optimization problems are modified to incorporate the cost-sensitive weights. This allows the algorithm to apply different penalties based on the class of the instance, making the updates more aggressive when misclassifying costly instances. The introduction of C_t to these algorithms ensures that the step size τ_t is scaled to reflect the cost associated with the class of the current instance. This cost-sensitive adjustment improves the algorithm's handling of imbalanced data and prioritizes the correction of costly misclassifications, enhancing its applicability in real-time CCPR:

The CSPA-I algorithm updates the step size using the following formula:

$$\tau_t = \min\left(C_t, \frac{\max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t))}{\|\mathbf{x}_t\|^2}\right) = \min\left(C_t, \frac{\ell(\mathbf{w}; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2}\right). \tag{3.25}$$

For the CSPA-II algorithm, the step size update is given by:

$$\tau_t = \frac{\max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t))}{\|\mathbf{x}_t\|^2 + \frac{1}{2C_t}} = \frac{\ell(\mathbf{w}; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2 + \frac{1}{2C_t}}. \tag{3.26}$$

In addition to the traditional cost-sensitive PA algorithms, six new CSPA variants are introduced to further refine the cost-sensitive learning process. These variants replace the hinge loss typically used in PA algorithms with the loss functions employed in CSOGD, namely L1 and L2 loss functions. The new variants: CSPA-L1, CSPA-L2 (derived from the PA algorithms), CSPA-I-L1, CSPA-I-L2 (derived from the PA-I algorithms), and CSPA-II-L1, CSPA-II-L2 (derived from the PA-II algorithms). These new variants incorporate the loss functions used in the CSOGD instead of the hinge loss. By replacing the hinge loss with these alternative loss functions, the variants aim to optimize performance in highly imbalanced data scenarios, where minimizing costly misclassifications is critical. Similar to CSPA-I and CSPA-II, these variants (except for those derived

from PA algorithms) also include class-specific penalty terms (C_t), and also solve the Passive-Aggressive optimization equations using the CSOGD loss to compute the new weight updates and step size τ_t.

For the L1 loss (borrowed from CSOGD-I), the weight update for CSPA-L1 remains similar to the original PA algorithm. The L1 loss function is defined as:

$$\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t)) = \max\left(0, (\rho \cdot \mathbb{I}_{y_t=1} + \mathbb{I}_{y_t=-1}) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right), \tag{3.27}$$

$$\mathbf{w} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t. \tag{3.28}$$

In CSPA-L1, the step size τ_t is computed as:

$$\tau_t = \frac{\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2}. \tag{3.29}$$

The CSPA-I-L1 algorithm modifies this step size calculation by incorporating the class-specific penalty term C_t:

$$\tau_t = \min\left(C_t, \frac{\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2}\right), \tag{3.30}$$

For CSPA-II-L1, the step size calculation includes an additional denominator term to further refine the updates:

$$\tau_t = \frac{\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2 + \frac{1}{2C_t}}. \tag{3.31}$$

The weight update rule and step size changes for the L2 loss function (borrowed from CSOGD-II),

$$\ell_{II}(\mathbf{w}; (\mathbf{x}_t, y_t)) = (\rho \cdot \mathbb{I}_{y_t=1} + \mathbb{I}_{y_t=-1}) \cdot \max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)). \tag{3.32}$$

$$\mathbf{w} = \mathbf{w}_t + \tau_t(\rho \cdot \mathbb{I}_{y_t=1} + \mathbb{I}_{y_t=-1}))y_t \mathbf{x}_t. \tag{3.33}$$

The corresponding step size calculation for CSPA-L2 is:

$$\tau_t = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{(\rho \cdot \mathbb{I}_{y_t=1} + \mathbb{I}_{y_t=-1})\|\mathbf{x}_t\|^2}. \tag{3.34}$$

The step sizes for CSPA-I-L2 and CSPA-II-L2 which incorporates the class-specific penalty term C_t are calculated below:

$$\tau_t = \min\left(C_t, \frac{\ell_{II}(\mathbf{w}; (\mathbf{x}_t, y_t))}{(\rho \cdot \mathbb{I}_{y_t=1} + \mathbb{I}_{y_t=-1})^2\|\mathbf{x}_t\|^2}\right). \tag{3.35}$$

$$\tau_t = \frac{\ell_{II}(\mathbf{w}; (\mathbf{x}_t, y_t))}{(\rho \cdot \mathbb{I}_{y_t=1} + \mathbb{I}_{y_t=-1})^2\|\mathbf{x}_t\|^2 + \frac{1}{2C_t}}. \tag{3.36}$$

Table 3.5: CSPA Algorithm Variants

Algorithm	**Loss Function**	**Class-Specific Penalties** (C^+ and C^-)
CSPA-I	Hinge Loss	Yes
CSPA-II	Hinge Loss	Yes
CSPA-L1	L1	None
CSPA-L2	L2	None
CSPA-I-L1	L1	Yes
CSPA-I-L2	L2	Yes
CSPA-II-L1	L1	Yes
CSPA-II-L2	L2	Yes

Algorithm 2 Passive-Aggressive (PA) and Cost-Sensitive Passive-Aggressive (CSPA) Algorithms

Input: Data stream $\{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$; Penalty parameter C; Class sizes n^+ and n^- (for CSPA variants); Loss function type $\ell_*(\mathbf{w}; (\mathbf{x}_t, y_t))$; Bias parameter ρ
Output: Final weight vector $\mathbf{w}_{T+1}$

Initialize weight vector $\mathbf{w}_1 = \mathbf{0}$
for $t = 1, \ldots, T$ **do** ▷ Iterate over each data point in the stream
  Receive instance $\mathbf{x}_t \in \mathbb{R}^n$ and label $y_t \in \{-1, +1\}$
  Predict: $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
  Compute loss $\ell_t(\mathbf{w}_t) = \ell_*(\mathbf{w}_t; (\mathbf{x}_t, y_t))$ where $* \in \{\text{Hinge}, I, II\}$ ▷ ℓ_* represents Hinge loss, L1, or L2 depending on the algorithm
  if $\ell_t(\mathbf{w}_t) > 0$ **then** ▷ Update only if loss is positive
    Compute penalty C_t where $* \in \{+, -, C\}$

$$C_t = \begin{cases} C & \text{for PA-I, PA-II,} \\ C^+ & \text{if } y_t = +1 \text{ in CSPA variants,} \\ C^- & \text{if } y_t = -1 \text{ in CSPA variants} \end{cases}$$

    Compute step size τ_t where $* \in \{\tau_{\text{PA}}, \tau_{\text{PA-I}}, \tau_{\text{PA-II}}\}$

$$\tau_t = \begin{cases} \tau_{\text{PA}}(\ell_t(\mathbf{w}_t)) & \text{for PA, CSPA-L1, CSPA-L2,} \\ \tau_{\text{PA-I}}(\ell_t(\mathbf{w}_t), C_t) & \text{for PA-I, CSPA-I, CSPA-I-L1, CSPA-I-L2,} \\ \tau_{\text{PA-II}}(\ell_t(\mathbf{w}_t), C_t) & \text{for PA-II, CSPA-II, CSPA-II-L1, CSPA-II-L2} \end{cases}$$

    Update weight vector:

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t + \tau_t y_t \mathbf{x}_t & \text{if Hinge or L1 loss,} \\ \mathbf{w}_t + \tau_t (\rho \cdot \mathbb{I}_{y_t=1} + \mathbb{I}_{y_t=-1}) y_t \mathbf{x}_t & \text{if L2 loss} \end{cases}$$

  else
    $\mathbf{w}_{t+1} = \mathbf{w}_t$ ▷ No update if loss is zero
  end if
end for
return $\mathbf{w}_{T+1}$

3.3.3 Performance Metrics and Experimental Setting

The experimental setup was designed to evaluate cost-sensitive online learning algorithms using synthetic data for CCPs. The experiments aimed to optimize the learning process under the constraints of imbalanced data, with a total of 1000 instances per dataset, comprising 100 positive samples ($T_p = n^+$) and 900 negative samples ($T_n = n^-$). This imbalance necessitates careful adjustment of the algorithms to appropriately weigh errors across classes.

Each dataset was generated as a unique combination of a window length w and an abnormal parameter value (g, s, a, d, σ'). A total of 380 unique combinations were created for each pattern, resulting in 380 distinct datasets. These combinations were used to simulate various scenarios of normal and abnormal patterns, thereby creating diverse training conditions. For example, a single dataset might represent an uptrend pattern with a window length $w = 10$ and a trend magnitude $g = 0.08$. Each of these datasets contained 1000 instances, maintaining the 100-to-900 ratio of positive to negative samples. For cyclic patterns, a period $\Omega = 8$ was used to capture periodic behaviors, reflecting the nature of specific abnormal data scenarios effectively.

The regularization parameter C, which controls the penalty for misclassification, was initialized to a default value of 1. For the learning rate λ, a time-decay approach was implemented, allowing it to decrease as more iterations are processed. Specifically, λ diminishes as the square root of the iteration number, promoting stability and convergence in the online learning scenario.

In cost-sensitive algorithms like CSOGD, the primary objective was to maximize the weighted sum of sensitivity (true positive rate) and specificity (true negative rate). The weights η_p and η_n were both set to 0.5, ensuring balanced consideration of false positives and false negatives throughout the training process. Given that CSOGD focuses on adjusting the model based on cost-sensitive classification without computing separate costs per class, the standard penalty parameter C is used directly in the algorithm's loss calculations. In contrast, the CSPA algorithms introduced class-specific costs C^+ and C^-, computed as $C^+ = C/n^+$ and $C^- = C/n^-$, to adjust for the imbalance between positive and negative instances.

To optimize the hyperparameters, the Optuna library was employed, which utilizes Bayesian optimization principles to efficiently search the parameter space. The regularization parameter C was tuned over a logarithmic range from 2^{-4} to 2^7 [8, 34], providing a broad exploration of possible values. The optimization process spanned 100 trials, where each trial involved training the model with a unique value of C and assessing its performance based on a chosen loss function. The optimal value of C, identified by Optuna, was then fixed and used consistently across multiple training runs to ensure robust performance.

The training process for all algorithms was conducted over 20 independent runs [36]. For each run, the entire dataset of 1000 instances was shuffled, ensuring a different sequence of data points and providing a robust evaluation of the model's learning capacity. The performance metrics, such as Accuracy, Sensitivity, Specificity, Precision, mistake count, G-Mean, Cost-Sensitive Sum (Weighted Sum of Sensitivity and Specificity), time, and Cumulative Error Rate (CER), were calculated at regular intervals or "ticks." The average values of these metrics across the 20 runs were recorded to offer a reliable assessment of each algorithm's performance.

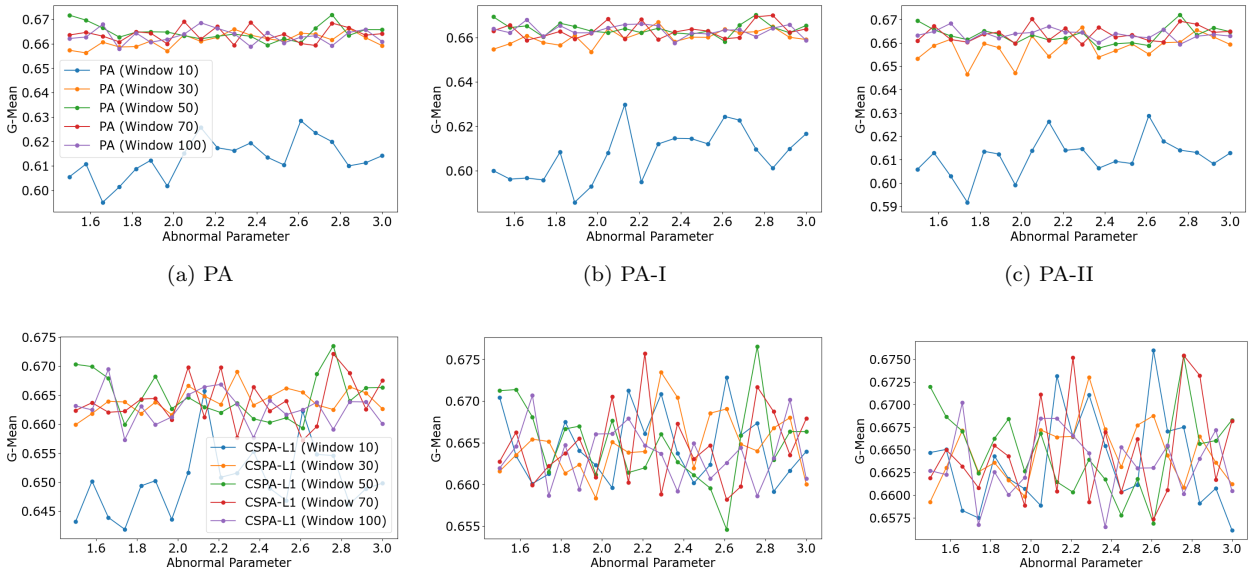
3.4 Computational Results

3.4.1 Results from Synthetic datasets

The computational experiments evaluate the performance of cost-sensitive and standard online learning algorithms across seven control chart patterns: *Uptrend, Downtrend, Upshift, Downshift, Systematic, Cyclic,* and *Stratification*. Fourteen algorithms were tested, including OGD, PA, PA-I, PA-II, and their cost-sensitive

counterparts: CSOGD-I, CSPA-L1, CSPA-I-L1, CSPA-II-L1 (using L1 loss), CSOGD-II, CSPA-L2, CSPA-I-L2, CSPA-II-L2 (using L2 loss), CSPA-I and CSPA-II. These experiments were implemented in Python, utilizing packages such as NumPy for numerical computations. The experiments were run on synthetic datasets using a high-performance computing cluster with parallelized code for efficient execution. The algorithms' performance was assessed based on metrics such as accuracy, G-Mean, weighted sum, and cost-efficiency, among others. Various plots and tables present the results, providing insight into each algorithm's behavior and effectiveness in detecting control chart patterns under different scenarios.

The comparison of standard online learning algorithms—OGD, PA, PA-I, and PA-II—across all control chart patterns reveals distinct differences in their performance. A notable finding is the significantly poorer performance of OGD when compared to the passive-aggressive variants. Across all patterns, including uptrend, downtrend, upshift, downshift, systematic, cyclic, and stratification, OGD struggles to maintain classification accuracy, showing instability and lower effectiveness. The passive-aggressive algorithms (PA, PA-I, and PA-II), on the other hand, exhibit much stronger classification results, particularly in terms of G-Mean and overall classification stability. While there are slight variations between PA, PA-I, and PA-II due to differences in their update mechanisms and step sizes, these differences do not translate into significant shifts in performance. The passive-aggressive models demonstrate superior resilience across varying window lengths and abnormal parameters, handling the complexity of control chart patterns with relative ease. Patterns with shorter window lengths or smaller abnormal parameters tend to be more challenging for all models, but passive-aggressive algorithms still outperform OGD consistently.



(a) PA (b) PA-I (c) PA-II

(d) CSPA-L1 (e) CSPA-I-L2 (f) CSPA-II-L2

Figure 3.6: Comparison of G-Mean of PA, PA-I, and PA-II among some cost-sensitive variants for Upshift patterns

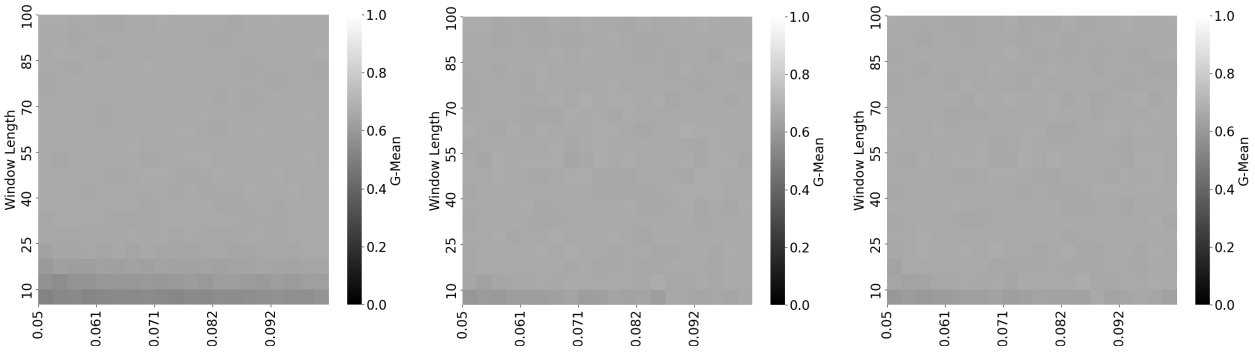
The cost-sensitive algorithms generally outperform their standard online learning counterparts (PA, PA-I, and PA-II) across multiple control chart patterns, particularly for smaller window lengths (e.g., $W = 10$). Models such as CSPA-L1, CSPA-I-L2, CSPA-II-L1, and CSPA-II-L2 demonstrate superior classification performance, overcoming the challenges faced by the original PA models, which exhibit a significant drop in G-Mean at smaller windows (Figure 3.6). However, while CSPA-L1 performs better than PA across all abnormal parameter values, it still struggles in classifying smaller window lengths compared to CSPA variants that incorporate class-specific penalties, such as CSPA-II-L2. This limitation arises because both CSPA-L1 and PA lack a regularization parameter C_t, meaning they do not utilize the cost-sensitive penalties C^+ and C^-. Experiments further reveal that algorithms equipped with both C^+ and C^- perform consistently better than those relying solely on L1 and L2 loss minimization without the additional penalty terms. While CSPA-I-L2 and CSPA-II-L2 leverage both loss functions and penalty terms to achieve higher G-Mean scores, CSPA-L1 lacks the flexibility provided by C^+ and C^-, limiting its ability to fully adapt to the imbalance between normal and abnormal patterns at shorter windows. Additional variants with L1 and L2 losses but without incorporating C^+ and C^- were tested, but they performed poorly and were excluded from the final study.

The trend analysis, based on increasing window lengths while keeping the abnormal parameter constant, reveals several key insights (Table 3.6). As the window length increases, classification performance improves gradually across all algorithms, with models incorporating both loss functions and penalty terms, such as CSPA-I-L2, CSPA-II-L1 and CSPA-II-L2, standing out at window lengths of 10 and 25. These results suggest

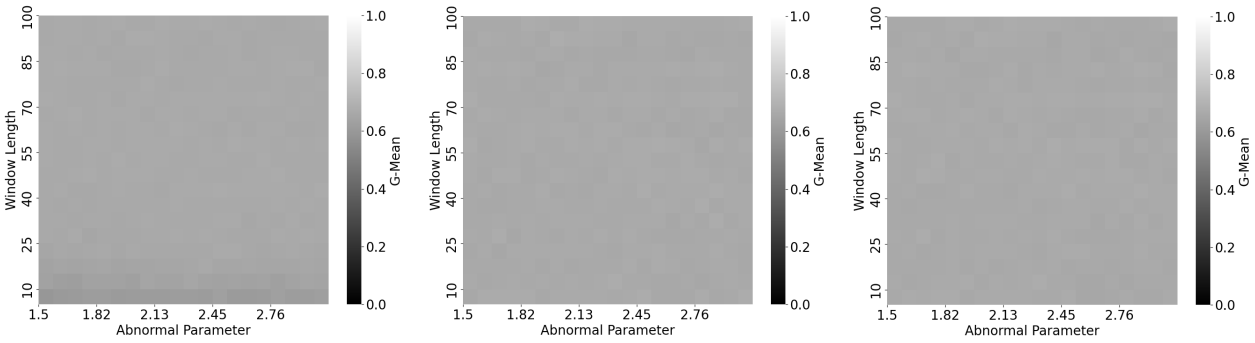
that the combination of class-specific penalties and optimized loss functions gives these algorithms a distinct advantage in detecting subtle changes. At window lengths above 50, the cost-sensitive algorithms continue to perform better; however, the G-Mean differences between cost-sensitive and standard online learning algorithms become marginal, indicating that longer windows provide sufficient contextual data to benefit all models similarly. A particularly interesting trend is the poor performance for stratification patterns, where many G-Mean scores drop below 50%. This could be due to the nature of stratification patterns, which lack a clear structure, making it harder for the algorithms to learn meaningful distinctions, or it could indicate that the chosen abnormal parameters did not fully capture the underlying variation. Additionally, cyclic patterns exhibit distinctive improvements with increasing window lengths, as cost-sensitive algorithms demonstrate sharper gains, suggesting that class-specific penalties effectively stabilize performance amidst periodic fluctuations. The heatmaps in Figure 3.7 provide a detailed visualization of the trends across different control chart patterns, further emphasizing the advantage of cost-sensitive algorithms at smaller window lengths and confirming that the inclusion of both loss functions and penalty terms enables better performance across varying window lengths and patterns, providing a robust approach to CCPR.

Table 3.6: G-Mean score of all algorithms across control chart patterns with four dataset variations

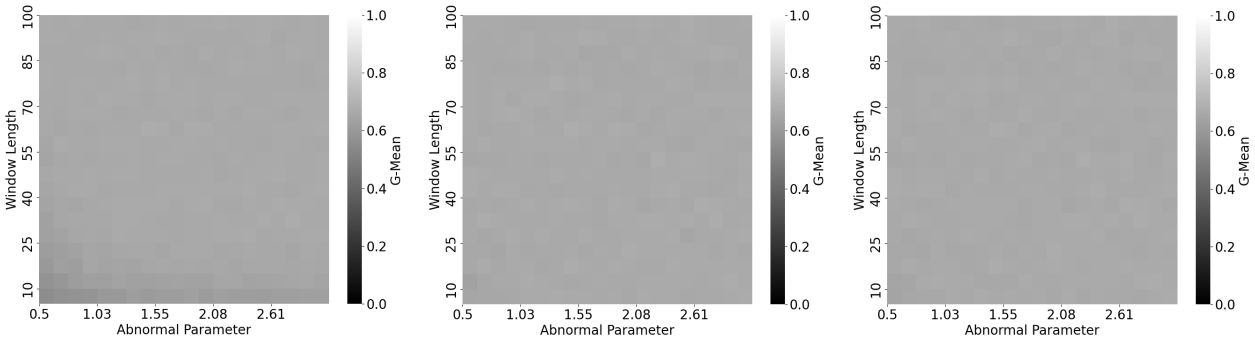
Pattern	w	abn	Algorithms													
			OGD	CSOGD-I	CSOGD-II	PA	CSPA-L1	CSPA-L2	PA-I	CSPA-I	CSPA-I-L1	CSPA-I-L2	PA-II	CSPA-II	CSPA-II-L1	CSPA-II-L2
Uptrend	10	g0.061	0.550	0.534	0.553	0.529	0.560	0.532	0.524	0.609	0.606	0.591	0.538	0.613	**0.616**	0.602
	25	g0.061	0.595	0.607	0.585	0.641	0.663	0.646	0.643	0.663	0.668	0.668	0.645	0.661	0.665	**0.669**
	50	g0.061	0.649	0.638	0.638	**0.662**	**0.662**	0.659	0.661	0.661	0.660	0.660	**0.662**	0.660	**0.662**	0.661
	75	g0.061	0.647	0.647	0.635	0.663	0.663	0.664	0.662	**0.664**	0.663	**0.664**	0.663	0.662	0.663	**0.664**
	95	g0.061	0.643	0.653	0.648	0.664	**0.666**	0.665	0.665	0.665	0.664	0.664	0.664	0.664	0.661	0.662
Downtrend	10	g0.05	0.505	0.506	0.500	0.523	0.534	0.519	0.504	0.583	0.589	0.592	0.519	0.579	**0.597**	0.579
	25	g0.05	0.636	0.636	0.623	0.649	0.667	0.647	0.643	0.674	0.673	**0.677**	0.642	0.664	**0.677**	0.676
	50	g0.05	0.656	0.643	0.645	0.666	0.668	0.667	0.666	0.668	0.670	**0.672**	0.669	0.668	0.671	**0.672**
	75	g0.05	0.663	0.670	0.658	0.672	0.674	0.672	0.671	**0.675**	0.673	0.673	0.670	0.673	0.673	0.671
	95	g0.05	0.638	0.632	0.644	0.661	0.663	0.663	0.662	0.661	0.660	**0.664**	0.661	0.661	0.662	0.662
Upshift	10	s1.5	0.636	0.643	0.652	0.605	0.643	0.605	0.600	0.669	0.669	**0.670**	0.606	0.664	**0.670**	0.665
	25	s1.5	0.603	0.634	0.629	0.651	0.663	0.652	0.650	**0.665**	0.664	0.663	0.654	0.662	0.661	0.663
	50	s1.5	0.633	0.656	0.653	0.672	0.670	0.671	0.669	**0.673**	**0.673**	0.671	0.669	0.672	**0.673**	0.672
	75	s1.5	0.668	0.664	0.661	0.676	**0.679**	0.675	0.675	0.678	0.678	**0.679**	0.672	0.675	0.678	**0.679**
	95	s1.5	0.642	0.651	0.643	0.667	0.666	0.666	0.668	0.667	0.668	0.668	0.666	0.667	**0.669**	**0.669**
Downshift	10	s2.21	0.647	0.642	0.652	0.619	0.654	0.618	0.616	0.666	0.668	0.667	0.620	**0.669**	0.668	0.668
	25	s2.21	0.614	0.613	0.569	0.656	**0.662**	0.657	0.657	0.661	0.659	0.660	0.656	0.659	0.659	0.661
	50	s2.21	0.625	0.624	0.622	0.662	0.661	0.660	0.661	0.661	0.661	**0.663**	0.661	0.661	0.660	0.660
	75	s2.21	0.650	0.650	0.655	0.668	0.667	0.667	0.667	0.669	0.667	0.668	0.668	**0.670**	0.668	0.667
	95	s2.21	0.646	0.628	0.622	0.664	0.666	0.665	0.665	0.663	0.665	0.665	0.666	0.664	**0.667**	**0.667**
Systematic	10	d0.89	0.594	0.580	0.523	0.584	0.629	0.583	0.588	0.657	0.658	**0.661**	0.581	0.658	0.659	0.659
	25	d0.89	0.629	0.638	0.628	0.652	0.666	0.653	0.642	0.665	0.666	0.666	0.650	0.666	0.666	**0.668**
	50	d0.89	0.600	0.621	0.612	0.664	0.664	0.663	0.661	0.664	0.665	0.664	0.663	**0.666**	0.663	0.664
	75	d0.89	0.618	0.603	0.596	0.662	0.661	**0.664**	0.663	0.660	0.662	0.661	0.663	0.662	0.663	0.662
	95	d0.89	0.635	0.629	0.614	0.660	**0.664**	0.663	0.662	0.662	0.663	0.660	0.659	0.660	0.663	0.663
Cyclic	10	a1.29	0.624	0.625	0.599	0.592	0.629	0.595	0.594	0.663	0.665	0.660	0.596	0.655	**0.667**	**0.667**
	25	a1.29	0.592	0.601	0.588	0.645	**0.662**	0.647	0.648	**0.662**	0.661	**0.662**	0.644	**0.662**	**0.662**	**0.662**
	50	a1.29	0.618	0.614	0.624	0.662	0.666	0.662	0.664	0.663	0.662	**0.667**	0.659	0.664	0.665	0.666
	75	a1.29	0.648	0.653	0.637	0.666	0.669	0.666	0.665	0.666	0.669	0.667	0.667	0.668	**0.671**	0.670
	95	a1.29	0.635	0.648	0.634	0.670	**0.671**	0.670	0.669	0.669	0.667	0.668	0.668	0.667	0.669	0.668
Stratification	10	$\sigma'0.35$	0.492	0.488	0.493	0.502	0.495	0.497	**0.508**	0.493	0.496	0.494	0.499	0.496	0.496	0.491
	25	$\sigma'0.35$	**0.505**	0.503	0.498	0.502	0.494	0.494	0.497	0.503	0.501	0.503	0.494	0.502	0.496	0.501
	50	$\sigma'0.35$	0.500	**0.508**	0.499	0.504	0.499	0.505	**0.508**	0.499	0.504	0.496	0.505	0.503	0.497	0.500
	75	$\sigma'0.35$	0.486	0.494	0.487	0.490	**0.497**	0.495	0.484	0.484	0.484	0.490	0.494	0.494	0.484	0.486
	95	$\sigma'0.35$	0.492	0.496	0.492	0.508	0.505	0.501	0.497	0.494	0.493	0.515	0.507	0.496	**0.515**	0.513



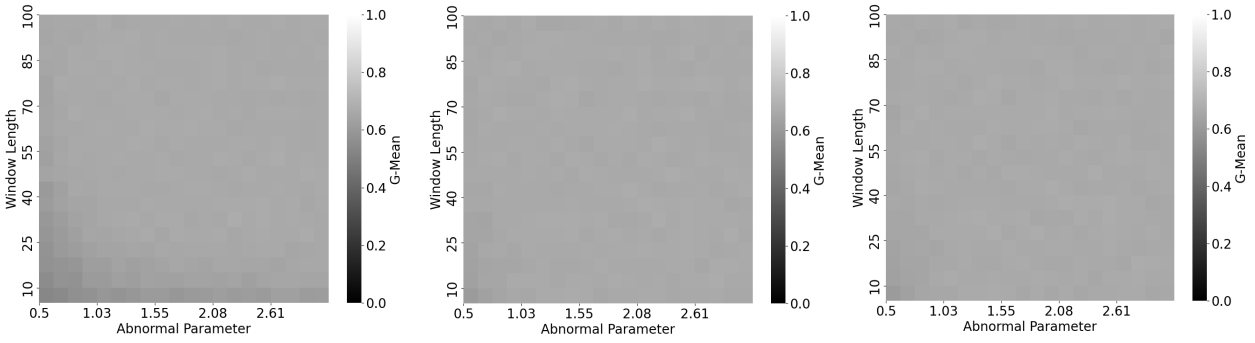
(a) Uptrend - PA (b) Uptrend - CSPA-II-L1 (c) Uptrend - CSPA-II-L2



(d) Downshift - PA (e) Downshift - CSPA-II-L1 (f) Downshift - CSPA-II-L2



(g) Systematic - PA (h) Systematic - CSPA-II-L1 (i) Systematic - CSPA-II-L2



(j) Cyclic - PA (k) Cyclic - CSPA-II-L1 (l) Cyclic - CSPA-II-L2

Figure 3.7: Heatmap comparison of PA, CSPA-II-L1, and CSPA-II-L2 algorithms across control chart patterns.

The results presented in Table 3.7 capture the Sum, Sensitivity, and Specificity metrics of cost-sensitive algorithms utilizing L1 and L2 loss functions, along with their associated standard deviations across multiple control chart patterns. The inclusion of ρ, as defined by Wang et al. [34], integrates the weights η_p and η_n (both set to 0.5), ensuring a balanced contribution of Sensitivity and Specificity to the Sum metric. This provides a comprehensive view of the algorithm's capacity to manage classification trade-offs between false positives and false negatives. A notable observation from the results is that algorithms such as CSOGD-I, CSOGD-II, CSPA-L1, and CSPA-L2 exhibit much higher standard deviations across the Sensitivity and Specificity metrics. This heightened variability suggests that these algorithms, which lack the incorporation of C^+ and C^- penalty terms, are more prone to instability under conditions with limited contextual data. The absence of class-specific penalties may cause these algorithms to respond more erratically to small fluctuations in data, amplifying inconsistencies across multiple runs. Conversely, the algorithms that incorporate C^+ and C^- penalty terms—such as CSPA-I-L1, CSPA-II-L1, CSPA-I-L2, and CSPA-II-L2—exhibit notably lower standard deviations. This indicates that the inclusion of class-specific penalties contributes to more stable performance, even when window lengths are small and contextual information is limited. These penalty terms help the algorithms better manage the trade-offs between different types of errors, reducing sensitivity to minor data variations. As a result, the algorithms with C^+ and C^- maintain consistent outcomes across multiple runs, ensuring reliable classification even under constrained conditions.

Table 3.8 captures the performance of the algorithms across various control chart patterns with a fixed window length of $w = 25$ and varying abnormal parameter values. Similar to the findings from Table 3.6, the algorithms incorporating both loss functions (L1 and L2) and class-specific penalty terms C^+ and C^- continue to outperform their counterparts, achieving the highest G-mean scores across multiple patterns. However, it is noteworthy that the performance of these algorithms does not show a consistent trend of improvement or decline with increasing abnormal parameter values. This suggests that while the incorporation of penalty terms enhances their robustness, the effectiveness of the algorithms does not necessarily scale with the magnitude of abnormal parameters. The fluctuating G-mean scores across different parameter values point to the inherent variability in performance, indicating that even with optimized loss functions and penalty mechanisms, the algorithms may encounter difficulties in maintaining stability as data characteristics shift.

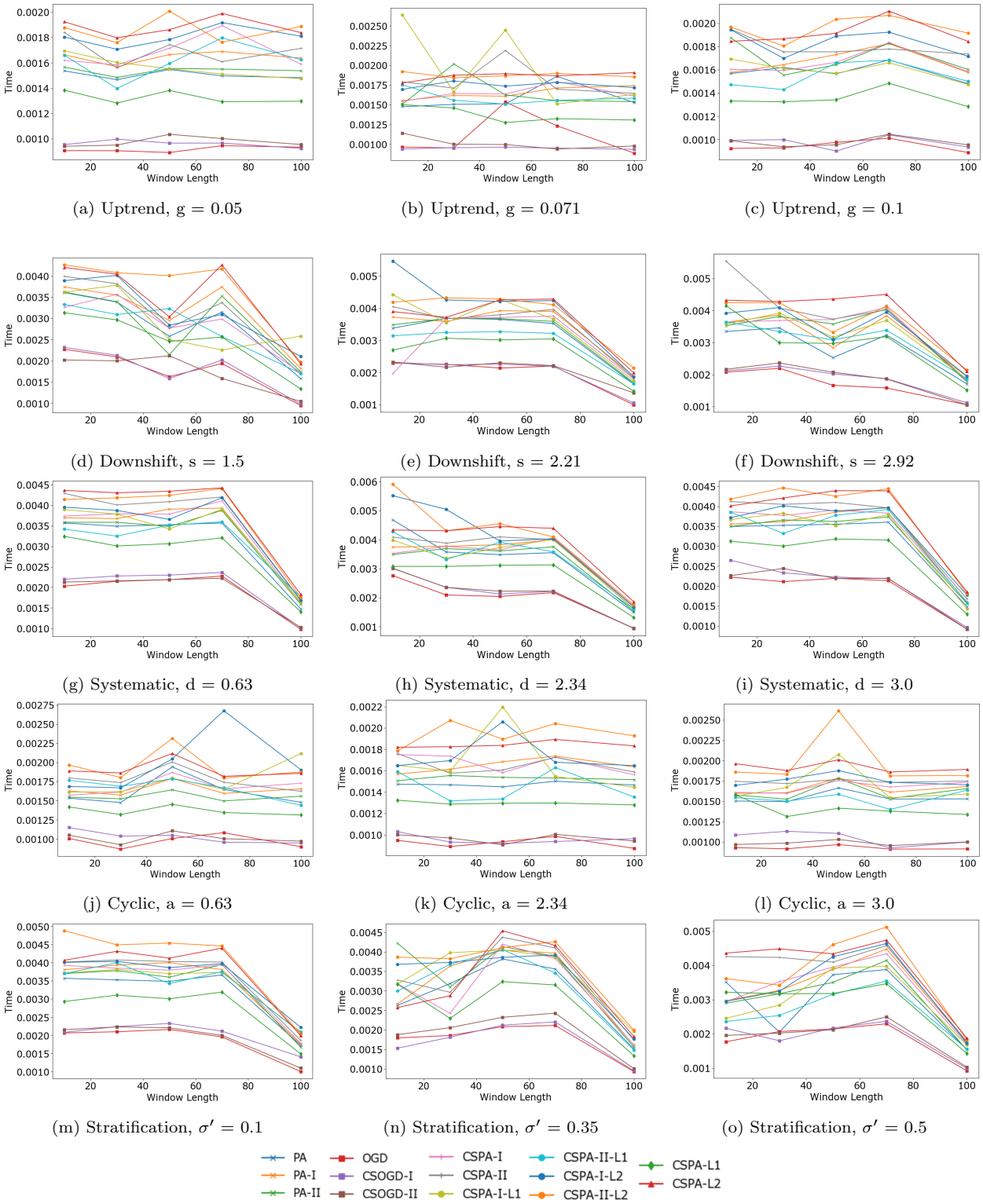
Table 3.7: Performance of all Algorithms utilizing L1 and L2 loss

Algorithm	"Uptrend", w = 15, g = 0.068			"Uptrend", w = 65, g = 0.076		
	Sum(%)	Sensitivity(%)	Specificity(%)	Sum(%)	Sensitivity(%)	Specificity(%)
CSOGD-I	60.999 ± 2.493	66.161 ± 3.658	55.836 ± 1.327	66.312 ± 0.989	73.939 ± 1.455	58.684 ± 0.525
CSOGD-II	57.491 ± 1.671	61.008 ± 2.456	53.973 ± 0.886	66.661 ± 1.105	74.464 ± 1.627	58.857 ± 0.584
CSPA-L1	63.260 ± 1.246	69.478 ± 1.834	57.042 ± 0.658	**67.467 ± 0.479**	**75.650 ± 0.722**	**59.283 ± 0.236**
CSPA-L2	60.305 ± 1.192	65.139 ± 1.753	55.471 ± 0.631	67.416 ± 0.512	75.575 ± 0.772	59.258 ± 0.252
CSPA-I-L1	66.604 ± 0.513	74.383 ± 0.753	58.824 ± 0.273	67.338 ± 0.369	75.458 ± 0.557	59.219 ± 0.182
CSPA-I-L2	66.938 ± 0.529	74.881 ± 0.788	58.995 ± 0.270	67.460 ± 0.218	75.644 ± 0.327	59.276 ± 0.110
CSPA-II-L1	**67.144 ± 0.438**	**75.186 ± 0.648**	**59.102 ± 0.229**	67.417 ± 0.190	75.575 ± 0.283	59.259 ± 0.096
CSPA-II-L2	67.033 ± 0.377	75.025 ± 0.563	59.041 ± 0.192	67.315 ± 0.533	75.419 ± 0.809	59.211 ± 0.258
Algorithm	"Downshift", w = 25, s = 1.58			"Downshift", w= 75, s = 2.76		
	Sum(%)	Sensitivity(%)	Specificity(%)	Sum(%)	Sensitivity(%)	Specificity(%)
CSOGD-I	64.517 ± 1.418	71.300 ± 2.078	57.733 ± 0.759	66.119 ± 1.383	73.647 ± 2.031	58.592 ± 0.736
CSOGD-II	62.781 ± 3.760	68.750 ± 5.508	56.811 ± 2.012	65.215 ± 2.943	72.339 ± 4.318	58.091 ± 1.569
CSPA-L1	66.995 ± 0.469	74.931 ± 0.718	59.059 ± 0.220	67.139 ± 0.439	75.153 ± 0.668	59.126 ± 0.211
CSPA-L2	66.151 ± 0.673	73.697 ± 1.007	58.604 ± 0.340	67.157 ± 0.546	75.175 ± 0.835	59.139 ± 0.258
CSPA-I-L1	**67.172 ± 0.312**	**75.206 ± 0.473**	**59.138 ± 0.151**	67.209 ± 0.320	75.258 ± 0.484	59.159 ± 0.156
CSPA-I-L2	67.061 ± 0.378	75.033 ± 0.580	59.088 ± 0.177	67.436 ± 0.226	75.611 ± 0.337	59.260 ± 0.114
CSPA-II-L1	66.971 ± 0.383	74.892 ± 0.585	59.051 ± 0.181	**67.462 ± 0.223**	**75.647 ± 0.336**	**59.276 ± 0.110**
CSPA-II-L2	67.033 ± 0.373	74.986 ± 0.568	59.081 ± 0.178	67.011 ± 0.470	74.958 ± 0.713	59.064 ± 0.228
Algorithm	"Systematic", w = 30, d= 0.76			"Systematic", w = 55, d = 2.87		
	Sum(%)	Sensitivity(%)	Specificity(%)	Sum(%)	Sensitivity(%)	Specificity(%)
CSOGD-I	57.836 ± 3.273	61.514 ± 4.808	54.158 ± 1.739	65.732 ± 2.739	73.086 ± 4.012	58.379 ± 1.466
CSOGD-II	58.876 ± 4.018	63.028 ± 5.888	54.724 ± 2.147	65.072 ± 1.858	72.131 ± 2.723	58.014 ± 0.993
CSPA-L1	66.656 ± 0.433	74.422 ± 0.658	58.890 ± 0.208	66.853 ± 0.514	74.714 ± 0.780	58.992 ± 0.248
CSPA-L2	65.734 ± 0.815	73.081 ± 1.216	58.387 ± 0.415	**66.895 ± 0.411**	**74.783 ± 0.625**	**59.007 ± 0.198**
CSPA-I-L1	66.617 ± 0.546	74.369 ± 0.815	58.864 ± 0.277	66.768 ± 0.316	74.581 ± 0.479	58.955 ± 0.152
CSPA-I-L2	66.856 ± 0.249	74.719 ± 0.380	58.993 ± 0.118	66.827 ± 0.335	74.681 ± 0.508	58.974 ± 0.163
CSPA-II-L1	66.797 ± 0.404	74.633 ± 0.615	58.961 ± 0.194	66.808 ± 0.424	74.647 ± 0.644	58.968 ± 0.204
CSPA-II-L2	**66.928 ± 0.276**	**74.828 ± 0.420**	**59.028 ± 0.132**	66.744 ± 0.331	74.547 ± 0.507	58.940 ± 0.156
Algorithm	"Cyclic", w = 10, a = 0.89			"Cyclic", w = 95, a = 2.74		
	Sum(%)	Sensitivity(%)	Specificity(%)	Sum(%)	Sensitivity(%)	Specificity(%)
CSOGD-I	60.329 ± 3.211	65.178 ± 4.716	55.480 ± 1.706	64.966 ± 2.078	71.939 ± 3.044	57.994 ± 1.113
CSOGD-II	59.427 ± 5.338	63.853 ± 7.844	55.001 ± 2.832	64.054 ± 3.833	70.608 ± 5.614	57.500 ± 2.052
CSPA-L1	61.474 ± 1.155	66.856 ± 1.704	56.092 ± 0.607	66.594 ± 0.399	74.311 ± 0.609	58.877 ± 0.190
CSPA-L2	57.959 ± 1.556	61.697 ± 2.288	54.221 ± 0.823	66.536 ± 0.509	74.222 ± 0.785	58.850 ± 0.235
CSPA-I-L1	65.741 ± 0.659	73.106 ± 0.971	58.377 ± 0.346	**66.608 ± 0.367**	**74.339 ± 0.556**	**58.877 ± 0.178**
CSPA-I-L2	65.925 ± 0.602	73.378 ± 0.898	58.473 ± 0.307	66.301 ± 0.474	73.856 ± 0.742	58.745 ± 0.208
CSPA-II-L1	66.052 ± 0.425	73.547 ± 0.629	58.556 ± 0.221	66.521 ± 0.304	74.200 ± 0.466	58.843 ± 0.144
CSPA-II-L2	**66.083 ± 0.524**	**73.614 ± 0.776**	**58.551 ± 0.273**	66.549 ± 0.347	74.253 ± 0.533	58.846 ± 0.161
Algorithm	"Stratification", w = 35, $\sigma' = 0.14$			"Stratification", w = 100, $\sigma' = 0.48$		
	Sum(%)	Sensitivity(%)	Specificity(%)	Sum(%)	Sensitivity(%)	Specificity(%)
CSOGD-I	49.773 ± 1.555	49.667 ± 2.286	49.880 ± 0.824	49.284 ± 1.834	48.947 ± 2.696	49.621 ± 0.973
CSOGD-II	50.654 ± 1.417	50.961 ± 2.084	50.346 ± 0.751	49.756 ± 1.550	49.642 ± 2.279	49.871 ± 0.822
CSPA-L1	50.434 ± 1.472	50.639 ± 2.165	50.230 ± 0.780	50.125 ± 1.532	50.183 ± 2.252	50.066 ± 0.812
CSPA-L2	**50.953 ± 1.685**	**51.400 ± 2.477**	**50.505 ± 0.893**	49.318 ± 1.658	48.997 ± 2.438	49.639 ± 0.879
CSPA-I-L1	50.345 ± 1.706	50.508 ± 2.508	50.182 ± 0.905	49.033 ± 1.349	48.578 ± 1.982	49.487 ± 0.715
CSPA-I-L2	50.361 ± 1.616	50.531 ± 2.376	50.192 ± 0.857	**50.236 ± 1.506**	**50.347 ± 2.214**	**50.125 ± 0.798**
CSPA-II-L1	50.177 ± 1.081	50.261 ± 1.589	50.094 ± 0.573	50.194 ± 1.271	50.286 ± 1.869	50.103 ± 0.673
CSPA-II-L2	50.521 ± 1.246	50.767 ± 1.832	50.276 ± 0.661	50.121 ± 1.308	50.178 ± 1.923	50.064 ± 0.693

Table 3.8: Performance of all algorithms across control chart patterns with varying abnormal parameter for $w = 25$

Pattern	Dataset	Algorithms													
		OGD	CSOGD-I	CSOGD-II	PA	CSPA-L1	CSPA-L2	PA-I	CSPA-I	CSPA-I-L1	CSPA-I-L2	PA-II	CSPA-II	CSPA-II-L1	CSPA-II-L2
Uptrend	g = 0.05	0.551	0.561	0.531	0.640	0.657	0.637	0.627	0.658	0.659	**0.661**	0.634	0.657	0.660	**0.661**
	g = 0.061	0.595	0.607	0.585	0.641	0.663	0.646	0.643	0.663	0.668	0.668	0.645	0.661	0.665	**0.669**
	g = 0.071	0.557	0.618	0.636	0.654	0.662	0.654	0.650	0.664	0.664	0.664	0.652	0.664	**0.666**	0.663
	g = 0.084	0.604	0.567	0.596	0.651	0.657	0.652	0.653	0.657	0.659	0.659	0.647	**0.660**	0.658	0.658
	g = 0.1	0.642	0.631	0.656	0.660	0.665	0.662	0.648	**0.667**	0.664	0.665	0.659	0.665	0.663	0.665
Downtrend	g = 0.05	0.636	0.636	0.623	0.649	0.667	0.647	0.643	0.674	0.673	**0.677**	0.642	0.664	**0.677**	0.676
	g = 0.061	0.594	0.584	0.598	0.648	0.660	0.647	0.648	0.662	0.660	0.662	0.648	0.661	**0.663**	0.661
	g = 0.071	0.573	0.577	0.578	0.647	**0.661**	0.648	0.647	0.658	0.659	0.660	0.648	0.660	0.660	**0.661**
	g = 0.084	0.643	0.611	0.643	0.660	0.665	0.658	0.659	0.666	0.665	0.666	0.659	0.663	**0.667**	0.666
	g = 0.1	0.633	0.626	0.618	0.657	**0.664**	0.658	0.648	0.663	0.661	0.663	0.654	0.663	0.663	0.662
Upshift	s = 1.5	0.603	0.634	0.629	0.651	0.663	0.652	0.650	**0.665**	0.664	0.663	0.654	0.662	0.661	0.663
	s = 1.89	0.618	0.614	0.590	0.652	0.660	0.652	0.652	0.660	0.662	0.661	0.653	0.656	0.662	**0.662**
	s = 2.21	0.639	0.625	0.644	0.661	0.668	0.663	0.663	0.669	0.673	0.668	0.660	0.666	**0.672**	0.671
	s = 2.68	0.656	0.646	0.654	0.664	0.667	0.663	0.663	0.671	0.670	0.668	0.658	0.668	**0.670**	0.668
	s = 2.92	0.630	0.610	0.635	0.655	0.660	0.655	0.654	0.658	0.657	**0.662**	0.657	0.660	0.656	**0.662**
Downshift	s = 1.5	0.598	0.614	0.585	0.649	**0.661**	0.650	0.650	0.659	0.658	0.660	0.652	0.659	0.659	0.660
	s = 1.89	0.631	0.631	0.613	0.655	0.663	0.657	0.656	0.662	0.661	0.663	0.655	0.662	0.662	**0.664**
	s = 2.21	0.614	0.613	0.569	0.656	**0.662**	0.657	0.657	0.661	0.659	0.660	0.656	0.659	0.659	0.661
	s = 2.68	0.640	0.628	0.622	0.656	0.660	0.655	0.656	0.661	0.661	0.660	0.652	0.660	0.660	**0.662**
	s = 2.92	0.640	0.644	0.635	0.657	0.662	0.657	0.656	0.663	0.660	0.662	0.655	0.663	**0.664**	0.662
Systematic	d = 0.63	0.562	0.567	0.573	0.628	0.650	0.626	0.626	0.651	0.659	0.663	0.626	0.649	**0.664**	0.653
	d = 0.89	0.629	0.638	0.628	0.652	0.666	0.653	0.642	0.665	0.666	0.666	0.650	0.666	0.666	**0.668**
	d = 1.29	0.637	0.642	0.626	0.656	0.667	0.655	0.658	0.662	0.662	0.663	0.656	0.662	0.664	**0.665**
	d = 1.95	0.651	0.656	0.655	0.663	0.667	0.661	0.662	0.668	0.668	0.671	0.660	0.665	**0.672**	**0.672**
	d = 2.34	0.645	0.627	0.652	0.661	0.663	0.658	0.645	0.662	**0.666**	0.664	0.661	0.663	0.665	0.662
Cyclic	a = 0.63	0.576	0.582	0.563	0.605	0.633	0.607	0.586	0.653	0.652	**0.656**	0.604	0.652	**0.656**	0.655
	a = 0.89	0.539	0.543	0.572	0.630	0.647	0.629	0.630	0.657	0.659	0.652	0.628	0.653	**0.661**	0.655
	a = 1.29	0.592	0.601	0.588	0.645	**0.662**	0.647	0.648	**0.662**	0.661	**0.662**	0.644	**0.662**	**0.662**	**0.662**
	a = 1.95	0.640	0.616	0.608	0.655	0.664	0.660	0.658	0.666	0.665	**0.668**	0.655	0.667	0.667	**0.668**
	a = 2.34	0.645	0.643	0.650	0.658	0.664	0.660	0.661	0.663	0.664	0.664	0.657	0.662	0.664	**0.665**
Stratification	$\sigma' = 0.1$	0.511	0.515	0.511	0.504	0.506	0.511	0.506	0.521	0.515	0.518	0.512	0.514	**0.524**	**0.524**
	$\sigma' = 0.21$	0.494	0.495	0.490	0.506	0.506	0.495	0.500	0.494	0.497	0.504	0.492	0.496	**0.513**	0.505
	$\sigma' = 0.35$	**0.505**	0.503	0.498	0.502	0.494	0.494	0.497	0.503	0.501	0.503	0.494	0.502	0.496	0.501
	$\sigma' = 0.44$	0.505	0.507	0.504	0.506	0.505	0.506	0.500	0.512	**0.517**	0.515	0.506	0.507	0.514	0.511
	$\sigma' = 0.5$	0.504	0.498	**0.505**	0.497	0.492	0.493	0.499	0.501	0.503	0.501	0.503	0.500	0.496	0.495

The computational time results in Figure 3.8 show a non-linear relationship with window length, where time remains relatively stable at smaller window lengths and only begins to drop significantly at larger window lengths (around $W = 70$ or more). This trend suggests that the algorithms might become more efficient in processing data as the window expands, possibly due to a reduction in noise sensitivity or optimized computational routines at higher window lengths. However, this behavior is not consistent across all patterns; notably, the uptrend pattern did not exhibit any significant drop in computational time, indicating that its structure might require similar computational resources regardless of window length. Furthermore, algorithms such as CSPA-I-L1, CSPA-I-L2, and CSPA-II-L2 exhibit significantly higher computational times across both window lengths and abnormal parameter variations. This increased time can be attributed to the complexity introduced by incorporating both L1 and L2 loss functions along with the cost-sensitive penalty terms C^+ and C^-, which demand additional computations at each iteration. CSPA-L2, despite lacking penalty terms, still incurs high computational overhead due to the use of dual loss functions.



(a) Uptrend, g = 0.05 (b) Uptrend, g = 0.071 (c) Uptrend, g = 0.1

(d) Downshift, s = 1.5 (e) Downshift, s = 2.21 (f) Downshift, s = 2.92

(g) Systematic, d = 0.63 (h) Systematic, d = 2.34 (i) Systematic, d = 3.0

(j) Cyclic, a = 0.63 (k) Cyclic, a = 2.34 (l) Cyclic, a = 3.0

(m) Stratification, $\sigma' = 0.1$ (n) Stratification, $\sigma' = 0.35$ (o) Stratification, $\sigma' = 0.5$

Figure 3.8: Comparison of Computational time versus window length across various abnormal patterns

Chapter 4: Conclusion

This study demonstrates the potential of advanced filtering techniques and cost-sensitive learning algorithms to enhance bioprocess monitoring in biopharmaceutical production. In the first part, we applied Kalman, particle, extended Kalman, and bilateral filters to impute missing recombinant protein titer data in *Escherichia coli* fermentation experiments. Using datasets from Cytovance Biologics, these filtering methods were evaluated, showing minimal differences in performance due to the high rate of missing data in *E. coli*. However, in a secondary dataset from Chinese Hamster Ovary (CHO) cell lines, bilateral filters exhibited the best performance, followed by Kalman filters. These findings lay the groundwork for integrating advanced filtering techniques into bioprocess optimization, offering the potential to improve production efficiency and reduce costs. Future research could focus on combining filtering techniques with machine learning models or developing hybrid models to further enhance predictive capabilities and titer estimation accuracy. Additionally, real-time integration of these techniques with bioprocess control systems could enable immediate interventions during fermentation, potentially reducing waste and increasing yields.

In the second part, we investigated the performance of cost-sensitive learning algorithms for CCPR. Our results showed that cost-sensitive models, such as the CSPA variants, consistently outperformed non-cost-sensitive models, especially when using class-specific penalties (C^+ and C^-) combined with L1 and L2 loss functions. The CSPA variants demonstrated superior performance in terms of G-Mean scores and were more effective at handling imbalanced datasets by balancing sensitivity and specificity. In particular, these models excelled in scenarios with smaller window lengths ($w \leq 25$), where standard Passive-Aggressive (PA) models exhibited significant performance drops. In contrast, OGD and its cost-sensitive variants (CSOGD-I and CSOGD-II) performed poorly across all control chart patterns, showing instability, low classification accuracy, compared to the PA and CSPA models. Even with the inclusion of cost-sensitive adjustments, CSOGD variants struggled with classification and error reduction, and their G-Mean scores remained consistently lower, indicating that OGD and its variants are ill-suited for handling complex control chart patterns.

CSPA models, particularly CSPA-L1, CSPA-I-L2, CSPA-II-L1, and CSPA-II-L2, maintained stability across various abnormal parameters and window lengths, showing superior G-Mean scores and remarkable stability in handling dynamic abnormal parameters. These models equipped with dual loss functions and class-specific penalties exhibited lower standard deviations in Sensitivity and Specificity, demonstrating more consistent outcomes across multiple runs. In contrast, models lacking these penalties exhibited higher standard deviations, reflecting their sensitivity to small fluctuations in data, leading to less stable performance. While the computational time for most CSPA variants was higher due to the complexity introduced by dual loss functions and class-specific penalties, the improved accuracy and stability of these models justify the increased processing time.

Despite the promising results, several limitations exist. In the case of the filtering techniques, their performance was hindered by the large volume of missing data in *E. coli* fermentations, which suggests a need for improved handling of high-missing-data scenarios in future studies. Additionally, the computational complexity of cost-sensitive learning algorithms, particularly at smaller window lengths, limits their real-time application in high-speed industrial environments. Future research could address these limitations by optimizing the computational efficiency of the algorithms and developing more robust methods for handling extensive missing data. Exploring alternative loss functions beyond L1 and L2, as well as fine-tuning class weight parameters (n_p and n_n), may further enhance adaptability in highly imbalanced scenarios. Expanding the range of abnormal parameters (g, s, a, d, σ') could also enable better detection of subtle and significant deviations in control chart patterns. Ultimately, these advancements would improve detection performance across a broader range of industrial applications.

References

[1] Abdoljalil Addeh, Aminollah Khormali, and Noorbakhsh Amiri Golilarz. "Control chart pattern recognition using RBF neural network with new training algorithm and practical features". In: *ISA transactions* 79 (2018), pp. 202–216.

[2] Zakaria Amribt et al. "Parameter identification for state estimation: Design of an extended Kalman filter for hybridoma cell fed-batch cultures". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 1170–1175.

[3] Monark Bag, Susanta Kumar Gauri, and Shankar Chakraborty. "An expert system for control chart pattern recognition". In: *The International Journal of Advanced Manufacturing Technology* 62 (2012), pp. 291–301.

[4] Yvo Boers and Johannes N Driessen. "Particle filter based detection for tracking". In: *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)*. Vol. 6. IEEE. 2001, pp. 4393–4397.

[5] Domenico Bonanni et al. "A Deep Learning Approach to Optimize Recombinant Protein Production in Escherichia coli Fermentations". In: *Fermentation* 9.6 (2023), p. 503.

[6] James Carpenter, Peter Clifford, and Paul Fearnhead. "Improved particle filter for nonlinear problems". In: *IEE Proceedings-Radar, Sonar and Navigation* 146.1 (1999), pp. 2–7.

[7] Catherine Ching Han Chang et al. "Periscope: quantitative prediction of soluble protein expression in the periplasm of Escherichia coli". In: *Scientific Reports* 6.1 (2016), p. 21844.

[8] Koby Crammer et al. "Online passive-aggressive algorithms." In: *Journal of Machine Learning Research* 7.3 (2006).

[9] Mohammad Derakhshi and Talayeh Razzaghi. "An imbalance-aware BiLSTM for control chart patterns early detection". In: *Expert Systems with Applications* 249 (2024), p. 123682.

[10] Laurent Dewasme et al. "Experimental validation of an Extended Kalman Filter estimating acetate concentration in E. coli cultures". In: *Journal of Process Control* 23.2 (2013), pp. 148–157.

[11] Petar M Djuric et al. “Particle filtering”. In: *IEEE Signal Processing Magazine* 20.5 (2003), pp. 19–38.

[12] Michael Elad. “On the origin of the bilateral filter and ways to improve it”. In: *IEEE Transactions on Image Processing* 11.10 (2002), pp. 1141–1151.

[13] Ethel Garcia et al. “Concurrent control chart pattern recognition: A systematic review”. In: *Mathematics* 10.6 (2022), p. 934.

[14] Ruturaj G Gavaskar and Kunal N Chaudhury. “Fast adaptive bilateral filtering”. In: *IEEE Transactions on Image Processing* 28.2 (2018), pp. 779–790.

[15] Thomas Gundinger et al. “Recombinant protein production in E. coli using the phoA expression system”. In: *Fermentation* 8.4 (2022), p. 181.

[16] Narjeskhatoon Habibi et al. “Prediction of recombinant protein overexpression in Escherichia coli using a machine learning based model (RPOLP)”. In: *Computers in Biology and Medicine* 66 (2015), pp. 330–336.

[17] Wafik Hachicha and Ahmed Ghorbel. “A survey of control-chart pattern-recognition literature (1991–2010) based on a new conceptual classification scheme”. In: *Computers & Industrial Engineering* 63.1 (2012), pp. 204–222.

[18] Chensong He, Jorge E Quijano, and Lisa M Zurk. “Enhanced Kalman filter algorithm using the invariance principle”. In: *IEEE Journal of Oceanic Engineering* 34.4 (2009), pp. 575–585.

[19] Steven CH Hoi et al. “Online learning: A comprehensive survey”. In: *Neurocomputing* 459 (2021), pp. 249–289.

[20] Cristovão Freitas Iglesias Jr et al. “Monitoring the Recombinant Adeno-Associated Virus Production using Extended Kalman Filter”. In: *Processes* 10.11 (2022), p. 2180.

[21] Andrew H Jazwinski. *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[22] Simon J Julier and Jeffrey K Uhlmann. “New extension of the Kalman filter to nonlinear systems”. In: *Signal Processing, Sensor Fusion, and Target Recognition VI*. Vol. 3068. Spie. 1997, pp. 182–193.

[23] Julian Kager, Christoph Herwig, and Ines Viktoria Stelzer. “State estimation for a penicillin fed-batch process combining particle filtering methods with online and time delayed offline measurements”. In: *Chemical Engineering Science* 177 (2018), pp. 234–244.

[24] Julian Kager et al. “Direct control of recombinant protein production rates in E. coli fed-batch processes by nonlinear feedback linearization”. In: *Chemical Engineering Research and Design* 182 (2022), pp. 290–304.

[25] Min Liu et al. “Metabolic engineering of Escherichia coli to improve recombinant protein production”. In: *Applied Microbiology and Biotechnology* 99 (2015), pp. 10367–10377.

[26] Christos P Papaneophytou and George Kontopidis. “Statistical approaches to maximize recombinant protein expression in Escherichia coli: a general review”. In: *Protein Expression and Purification* 94 (2014), pp. 22–32.

[27] Sylvain Paris and Frédo Durand. “A fast approximation of the bilateral filter using a signal processing approach”. In: *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part IV 9*. Springer. 2006, pp. 568–580.

[28] PR Patnaik. “An integrated hybrid neural system for noise filtering, simulation and control of a fed-batch recombinant fermentation”. In: *Biochemical Engineering Journal* 15.3 (2003), pp. 165–175.

[29] Sha Sha et al. “Mechanistic modeling and applications for CHO cell culture development and production”. In: *Current Opinion in Chemical Engineering* 22 (2018), pp. 54–61.

[30] R Simutis et al. “State estimation of a biotechnological process using extended Kalman filter and Particle filter”. In: *Veterinary and Agricultural Engineering* (2014), pp. 920–924.

[31] Arne Skerra and Andreas Plückthun. “Assembly of a functional immunoglobulin Fv fragment in Escherichia coli”. In: *Science* 240.4855 (1988), pp. 1038–1041.

[32] Carlo Tomasi and Roberto Manduchi. “Bilateral filtering for gray and color images”. In: *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE. 1998, pp. 839–846.

[33] Ramazan Ünlü. “A robust data simulation technique to improve early detection performance of a classifier in control chart pattern recognition systems”. In: *Information Sciences* 548 (2021), pp. 18–36.

[34] Jialei Wang, Peilin Zhao, and Steven CH Hoi. “Cost-sensitive online classification”. In: *IEEE Transactions on Knowledge and Data Engineering* 26.10 (2013), pp. 2425–2438.

[35] Jialei Wang, Peilin Zhao, and Steven CH Hoi. “Exact soft confidence-weighted learning”. In: *arXiv preprint arXiv:1206.4612* (2012).

[36] Jialei Wang, Peilin Zhao, and Steven CH Hoi. “Libol: A library for online learning algorithms”. In: (2014).

[37] Greg Welch, Gary Bishop, et al. “An introduction to the Kalman filter”. In: (1995).

[38] Petros Xanthopoulos and Talayeh Razzaghi. “A weighted support vector machine method for control chart pattern recognition”. In: *Computers & Industrial Engineering* 70 (2014), pp. 134–149.

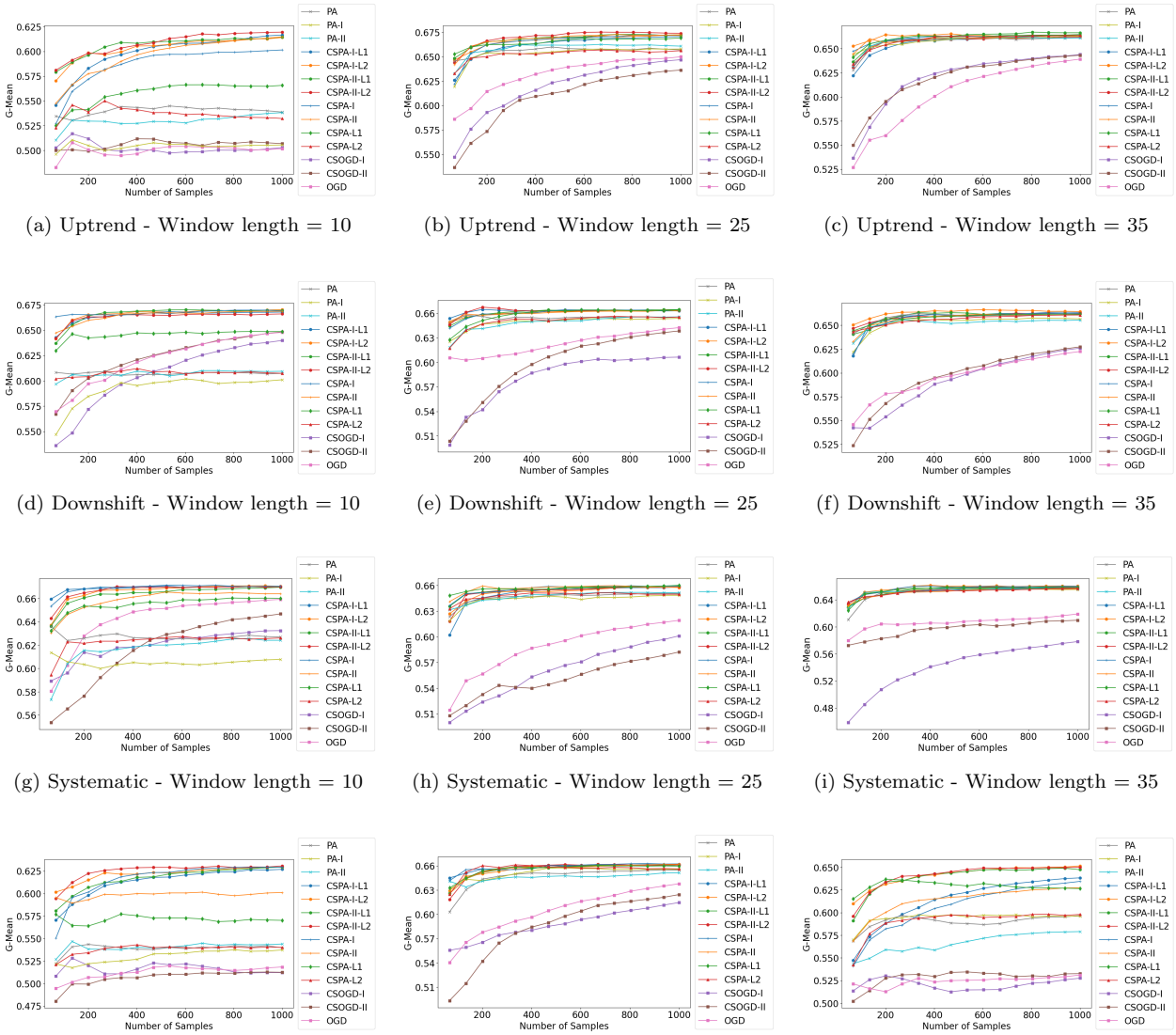
[39] Li Xue et al. “Control chart pattern recognition for imbalanced data based on multi-feature fusion using convolutional neural network”. In: *Computers & Industrial Engineering* 182 (2023), p. 109410.

[40] Qingxiong Yang. “Recursive bilateral filtering”. In: *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I 12*. Springer. 2012, pp. 399–413.

Appendices

Plots

Smaller Window Lengths



(a) Uptrend - Window length = 10 (b) Uptrend - Window length = 25 (c) Uptrend - Window length = 35

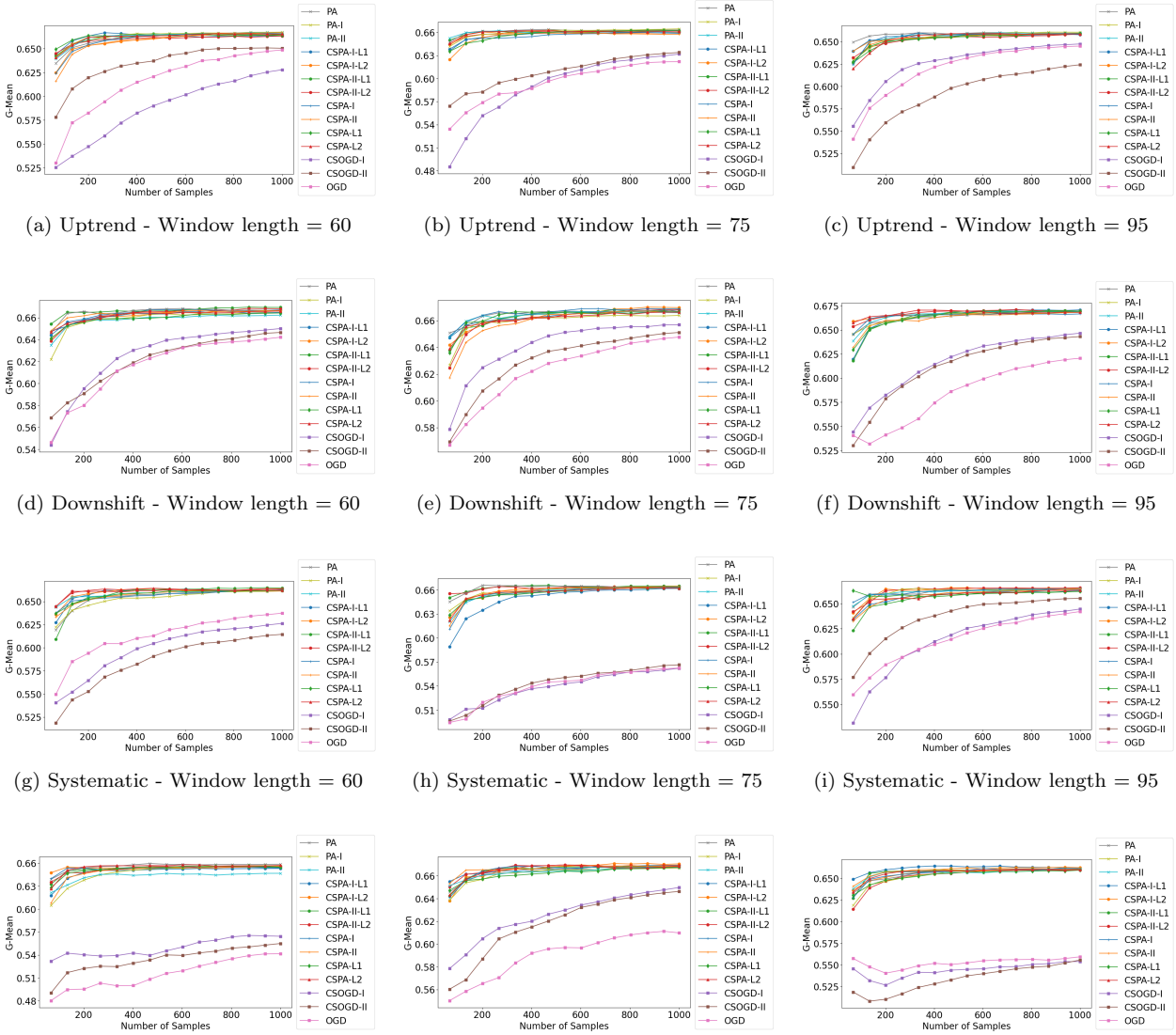
(d) Downshift - Window length = 10 (e) Downshift - Window length = 25 (f) Downshift - Window length = 35

(g) Systematic - Window length = 10 (h) Systematic - Window length = 25 (i) Systematic - Window length = 35

(j) Cyclic - Window length = 10 (k) Cyclic - Window length = 25 (l) Cyclic - Window length = 35

Figure 4.9: G-mean for small window lengths across different patterns.

Larger Window Lengths



(a) Uptrend - Window length = 60 (b) Uptrend - Window length = 75 (c) Uptrend - Window length = 95

(d) Downshift - Window length = 60 (e) Downshift - Window length = 75 (f) Downshift - Window length = 95

(g) Systematic - Window length = 60 (h) Systematic - Window length = 75 (i) Systematic - Window length = 95

(j) Cyclic - Window length = 60 (k) Cyclic - Window length = 75 (l) Cyclic - Window length = 95

Figure 4.10: G-mean for large window lengths across different patterns.

PA, PA-I and PA-II for L1 loss

Solve PA for L1 loss

The objective is to minimize:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2,$$

subject to:

$$\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0,$$

where the loss function is defined as:

$$\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t)) = \max(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w} \cdot \mathbf{x}_t)).$$

Step 1: Lagrangian Formulation

The Lagrangian is formulated to handle the constraint:

$$\mathcal{L}(\mathbf{w}, \xi, \tau, \lambda) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + \tau \left(\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t))\right),$$

$$\mathcal{L}(\mathbf{w}, \xi, \tau, \lambda) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + \tau \left(\max(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w} \cdot \mathbf{x}_t))\right),$$

Step 2: Differentiate w.r.t. $\mathbf{w}$

Differentiate the Lagrangian w.r.t. $\mathbf{w}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \mathbf{w}_t - \tau y_t \mathbf{x}_t = 0.$$

Solving for $\mathbf{w}$:

$$\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t.$$

Substituting $\mathbf{w}$ into the Lagrangian, we get:

$$\mathcal{L}(\tau) = \frac{1}{2}\tau^2\|\mathbf{x}_t\|^2 + \tau\left(\max\left(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau\|\mathbf{x}_t\|^2\right)\right).$$

The Lagrangian is:

$$\mathcal{L}(\tau) = \frac{1}{2}\tau^2\|\mathbf{x}_t\|^2 + \tau \left(\max\left(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau\|\mathbf{x}_t\|^2\right)\right).$$

Step 1: Consider the case when the max term is non-zero When the term inside the max is positive, the Lagrangian simplifies to:

$$\mathcal{L}(\tau) = \frac{1}{2}\tau^2\|\mathbf{x}_t\|^2 + \tau \left((\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau\|\mathbf{x}_t\|^2\right).$$

Step 2: Expand the Lagrangian

$$\mathcal{L}(\tau) = -\frac{1}{2}\tau^2\|\mathbf{x}_t\|^2 + \tau \left((\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right).$$

Step 3: Differentiate w.r.t. τ

$$\frac{d\mathcal{L}}{d\tau} = -\tau\|\mathbf{x}_t\|^2 + \left((\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right).$$

Step 4: Solve for τ Set the derivative equal to zero:

$$-\tau\|\mathbf{x}_t\|^2 + \left((\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right) = 0,$$

which gives:

$$\tau = \frac{(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2}.$$

Solve PA-I for L1 loss

The objective is to minimize:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi,$$

subject to:

$$\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t)) \le \xi, \quad \xi \ge 0,$$

where the loss function is defined as:

$$\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t)) = \max(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w} \cdot \mathbf{x}_t)).$$

Step 1: Lagrangian Formulation

The Lagrangian is formulated to handle the constraint:

$$\mathcal{L}(\mathbf{w}, \xi, \tau, \lambda) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi + \tau\left(\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t)) - \xi\right) - \lambda(\xi),$$

Step 2: Differentiate w.r.t. $\mathbf{w}$

Differentiate the Lagrangian w.r.t. $\mathbf{w}$:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \mathbf{w} - \mathbf{w}_t - \tau y_t \mathbf{x}_t = 0.$$

Solving for $\mathbf{w}$:

$$\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t.$$

Step 3: Differentiate w.r.t. ξ

Now, differentiate the Lagrangian w.r.t. ξ:

$$\frac{\partial \mathcal{L}}{\partial \xi} = C - \tau - \lambda = 0.$$

Solving for λ:

$$\lambda = C - \tau.$$

Step 4: Substitute Exact Loss and Expand Lagrangian

Substitute $\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$ and the exact loss expression into the Lagrangian:

- The loss function is:

$$\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t)) = \max\left(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w} \cdot \mathbf{x}_t)\right).$$

- Substitute $\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$:

$$\ell_I = \max\left(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t\left((\mathbf{w}_t + \tau y_t \mathbf{x}_t) \cdot \mathbf{x}_t\right)\right).$$

- Expand the dot product:

$$\ell_I = \max\left(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau\|\mathbf{x}_t\|^2\right).$$

The Lagrangian becomes:

$$\mathcal{L}(\tau) = \frac{1}{2}\tau^2\|\mathbf{x}_t\|^2 + C\xi + \tau\left(\max\left(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau\|\mathbf{x}_t\|^2\right) - \xi\right) - (C - \tau)\xi.$$

Step 5: Simplify and Differentiate w.r.t. τ

Now, expand and simplify the Lagrangian:

$$\mathcal{L}(\tau) = \frac{1}{2}\tau^2\|\mathbf{x}_t\|^2 + \tau\left(\max\left(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right)\right) - \tau^2\|\mathbf{x}_t\|^2 - \tau\xi - (C - \tau)\xi.$$

Differentiate w.r.t. τ:

$$\frac{\partial \mathcal{L}}{\partial \tau} = -\tau\|\mathbf{x}_t\|^2 + \max\left(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right).$$

Set this derivative to zero:

$$-\tau\|\mathbf{x}_t\|^2 + \ell_I = 0 \implies \tau = \frac{\ell_I}{\|\mathbf{x}_t\|^2}.$$

Step 6: Use KKT Conditions

Since τ should be non-negative and bounded by C, apply the KKT conditions:

$$\tau_t = \min\left(C, \frac{\ell_I}{\|\mathbf{x}_t\|^2}\right),$$

where:

$$\ell_I = \max\left(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right).$$

Step 7: Weight Update Rule

The final weight update rule is:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t.$$

Solve PA-II for L1 loss

To solve the optimization problem given by:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 \quad \text{subject to} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \le \xi,$$

Step 1: Constraint Simplification

Recall the loss function:

$$\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t)) = \max(0, (\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w} \cdot \mathbf{x}_t)).$$

The constraint:

$$\ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \le \xi$$

implies:

$$(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w} \cdot \mathbf{x}_t) \le \xi.$$

Step 2: Optimization with Lagrange Multiplier

The objective function to minimize is:

$$\frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2.$$

The constraint introduces a Lagrange multiplier λ to form the Lagrangian:

$$\mathcal{L}(\mathbf{w}, \xi, \tau) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 + \tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w} \cdot \mathbf{x}_t) - \xi\right].$$

Step 3: Solving for $\mathbf{w}$

To find the optimal $\mathbf{w}$, take the derivative of $\mathcal{L}$ with respect to $\mathbf{w}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \mathbf{w}_t - \tau y_t \mathbf{x}_t = 0.$$

Solving this equation, we find:

$$\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t.$$

Step 4: Solving for ξ

Now take the derivative of $\mathcal{L}$ with respect to ξ:

$$\frac{\partial \mathcal{L}}{\partial \xi} = 2C\xi - \tau = 0.$$

Solving for ξ:

$$\xi = \frac{\tau}{2C}.$$

Step 5: Substitute $\mathbf{w}$ and ξ into the Lagrangian

The Lagrangian function is:

$$\mathcal{L}(\mathbf{w}, \xi, \tau) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 + \tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w} \cdot \mathbf{x}_t) - \xi\right].$$

Substitute the expressions for $\mathbf{w}$ and ξ: $\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$, $\xi = \frac{\tau}{2C}$.

First term:

$$\frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 = \frac{1}{2}\|\tau y_t \mathbf{x}_t\|^2 = \frac{1}{2}\tau^2\|\mathbf{x}_t\|^2.$$

Second term:

$$C\xi^2 = C\left(\frac{\tau}{2C}\right)^2 = \frac{\tau^2}{4C}.$$

Third term:

$$\tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau y_t^2 \|\mathbf{x}_t\|^2 - \frac{\tau}{2C}\right].$$

Step 6: Form the Full Lagrangian

Now, differentiate $\mathcal{L}$ w.r.t. τ:

$$\frac{\partial \mathcal{L}}{\partial \tau} = \tau\|\mathbf{x}_t\|^2 + \frac{\tau}{2C} + \left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right] - 2\tau\|\mathbf{x}_t\|^2 - \frac{\tau}{C}.$$

Simplifying the derivative:

$$\frac{\partial \mathcal{L}}{\partial \tau} = -\tau\|\mathbf{x}_t\|^2 - \frac{\tau}{2C} + \left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right].$$

$$-\tau\|\mathbf{x}_t\|^2 - \frac{\tau}{2C} + \left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right] = 0.$$

Rearranging terms:

$$-\tau\|\mathbf{x}_t\|^2 - \frac{\tau}{2C} = -\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right].$$

Factoring out τ:

$$\tau\left(\|\mathbf{x}_t\|^2 + \frac{1}{2C}\right) = \left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\right].$$

Solving for τ:

$$\tau = \frac{(\rho \cdot I(y_t = 1) + I(y_t = -1)) - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}}.$$

$$\tau = \frac{\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}}.$$

PA, PA-I and PA-II for L2 loss

Solve PA for L2 loss

The objective is to minimize:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2,$$

subject to:

$$\ell_I(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0,$$

where the loss function is defined as:

$$\ell_{II}(\mathbf{w};(\mathbf{x}_t,y_t)) = (\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t)).$$

Step 1: Lagrangian Formulation

The Lagrangian is formulated to handle the constraint:

$$\mathcal{L}(\mathbf{w}, \tau,) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + \tau\left(\ell_I(\mathbf{w};(\mathbf{x}_t,y_t))\right),$$

The Lagrangian for this constrained problem is:

$$\mathcal{L}(\mathbf{w}, \tau,) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + \tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t))\right].$$

Step 2: Solving for $\mathbf{w}$

To find $\mathbf{w}$, differentiate the Lagrangian with respect to $\mathbf{w}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \mathbf{w}_t - \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t = 0.$$

Solving for $\mathbf{w}$:

$$\mathbf{w} = \mathbf{w}_t + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t.$$

Step 3: Substitute $\mathbf{w}$ into the Lagrangian

Substituting $\mathbf{w}$ into the Lagrangian:

$$\|\mathbf{w} - \mathbf{w}_t\|^2 = \tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2.$$

Substituting $\mathbf{w} = \mathbf{w}_t + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t$:

$$\|\mathbf{w} - \mathbf{w}_t\|^2 = \tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2.$$

The loss function becomes:

$$\max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))\|\mathbf{x}_t\|^2).$$

Thus, the final Lagrangian is:

$$\mathcal{L}(\tau) = \frac{1}{2}\tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + \tau\Bigg[(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max\Big(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))\|\mathbf{x}_t\|^2\Big)\Bigg].$$

The Lagrangian is given as:

$$\mathcal{L}(\tau) = \frac{1}{2}\tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + \tau\Bigg[(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max\Big(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))\|\mathbf{x}_t\|^2\Big)\Bigg].$$

Step 1: Simplify the max term Assuming the max term is non-negative, we simplify it to:

$$\mathcal{L}(\tau) = -\frac{1}{2}\tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)).$$

Step 2: Differentiate with respect to τ

$$\frac{d\mathcal{L}}{d\tau} = -\tau(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + (\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)).$$

Step 3: Solve for τ Set the derivative equal to zero:

$$\tau = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{(\rho \cdot I(y_t = 1) + I(y_t = -1))\|\mathbf{x}_t\|^2}.$$

Solve PA-I for L2 loss

The objective is to minimize:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi,$$

subject to:

$$\ell_{II}(\mathbf{w}; (\mathbf{x}_t, y_t)) \le \xi, \quad \xi \ge 0,$$

where the loss function is:

$$\ell_{II}(\mathbf{w}; (\mathbf{x}_t, y_t)) = (\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t)).$$

Step 1: Formulating the Lagrangian

The Lagrangian for this constrained problem is:

$$\mathcal{L}(\mathbf{w}, \xi, \tau, \lambda) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi + \tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t)) - \xi\right] - \lambda\xi.$$

Step 2: Solving for $\mathbf{w}$

To find $\mathbf{w}$, differentiate the Lagrangian with respect to $\mathbf{w}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \mathbf{w}_t - \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t = 0.$$

Solving for $\mathbf{w}$:

$$\mathbf{w} = \mathbf{w}_t + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t.$$

Step 3: Solving for ξ

Now, differentiate the Lagrangian with respect to ξ:

$$\frac{\partial \mathcal{L}}{\partial \xi} = C - \tau - \lambda = 0.$$

Solving for λ:

$$\lambda = C - \tau.$$

Step 4: Substitute $\mathbf{w}$ and ξ into the Lagrangian

We are given the Lagrangian function:

$$\mathcal{L}(\mathbf{w}, \xi, \tau, \lambda) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi + \tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t)) - \xi\right] - \lambda\xi.$$

We know:

$$\mathbf{w} = \mathbf{w}_t + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t,$$

and

$$\lambda = C - \tau.$$

Substituting these into the Lagrangian:

$$\mathcal{L}(w, \tau, \lambda, \xi) = \frac{1}{2}\|\mathbf{w}_t + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t - \mathbf{w}_t\|^2 + C\xi$$
$$+\tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t\left((\mathbf{w}_t + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t) \cdot \mathbf{x}_t\right)) - \xi\right]$$
$$-(C - \tau)\xi.$$

Now, substituting $\mathbf{w} = \mathbf{w}_t + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t$ and $\lambda = C - \tau$:

$$\mathcal{L}(\tau, \xi) = \frac{1}{2}\|\tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t\|^2 + C\xi$$
$$+\tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t\left((\mathbf{w}_t + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t) \cdot \mathbf{x}_t\right)) - \xi\right]$$
$$-(C - \tau)\xi.$$

Now, expanding this expression:

$$\mathcal{L} = \frac{1}{2}\tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + C\xi$$
$$+\tau\left((\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t\mathbf{w}_t \cdot \mathbf{x}_t) - \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 - \xi\right)$$
$$-(C - \tau)\xi.$$

After substituting and simplifying further:

$$= \frac{1}{2}\tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + \xi C$$
$$+\left[\tau(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t\mathbf{w}_t \cdot \mathbf{x}_t) - \tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 - \tau\xi\right]$$
$$-C\xi + \tau\xi.$$

Taking the derivative with respect to τ:

$$\frac{\partial\mathcal{L}}{\partial\tau} = -\tau(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + (\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)) - \xi.$$

Setting this to 0:

$$\tau(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 = (\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)) - \xi.$$

Thus, solving for τ:

$$\tau = \frac{(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot (1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t))}{(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2}.$$

Finally, the solution becomes:

$$\tau = \frac{\ell_{II}}{(\rho \cdot I_{y_t=1} + I_{y_t=-1})^2\|\mathbf{x}_t\|^2}$$

Thus, for the case where $\tau < C$ or $\frac{\ell_{II}}{(\rho \cdot I_{y_t=1} + I_{y_t=-1})^2 \|\mathbf{x}_t\|^2} \leq C$, we have the final expression for τ:

$$\tau = \min\left(C, \frac{\ell_{II}}{(\rho \cdot I_{y_t=1} + I_{y_t=-1})^2 \|\mathbf{x}_t\|^2}\right).$$

Solve PA-II for L2 loss

To solve the optimization problem given by:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 \quad \text{subject to} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi,$$

we will proceed as follows using the new loss function.

New Loss Function where the loss function is:

$$\ell_{II}(\mathbf{w}; (\mathbf{x}_t, y_t)) = (\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t)).$$

Step 1: Formulating the Lagrangian

The Lagrangian function is:

$$\mathcal{L}(\mathbf{w}, \xi, \tau) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 + \tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t)) - \xi\right],$$

where τ is the Lagrange multiplier.

Step 2: Solving for $\mathbf{w}$

Take the derivative of the Lagrangian with respect to $\mathbf{w}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \mathbf{w}_t - \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t = 0.$$

Solving for $\mathbf{w}$:

$$\mathbf{w} = \mathbf{w}_t + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t.$$

Step 3: Solving for ξ

Take the derivative of the Lagrangian with respect to ξ:

$$\frac{\partial \mathcal{L}}{\partial \xi} = 2C\xi - \tau = 0.$$

Solving this equation for ξ:

$$\xi = \frac{\tau}{2C}.$$

Step 4: Substitute $\mathbf{w}$ and ξ into the Lagrangian

The Lagrangian function is:

$$\mathcal{L}(\mathbf{w}, \xi, \tau) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 + \tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t)) - \xi\right].$$

Substitute the expressions for $\mathbf{w}$ and ξ: $\mathbf{w} = \mathbf{w}_t + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))y_t\mathbf{x}_t$,

$\xi = \frac{\tau}{2C}$.

Expanding the Terms

1. **First term**: $\frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2$:

$$\frac{1}{2}\tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2.$$

2. **Second term**: $C\xi^2$:

$$C\left(\frac{\tau}{2C}\right)^2 = \frac{\tau^2}{4C}.$$

3. **Third term**: $\tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1)) \cdot \max\left(0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))\|\mathbf{x}_t\|^2\right) - \xi\right]$:

$$\tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1))\left(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t) - \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))\|\mathbf{x}_t\|^2\right) - \frac{\tau}{2C}\right].$$

Step 5: Form the Full Lagrangian

Putting all the terms together, the Lagrangian becomes:

$$\mathcal{L}(\tau) = \frac{1}{2}\tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + \frac{\tau^2}{4C}$$

$$+\tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)) - \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 - \frac{\tau}{2C}\right].$$

Expanding and simplifying:

$$\mathcal{L}(\tau) = \frac{1}{2}\tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 - \tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + \frac{\tau^2}{4C} - \frac{\tau^2}{2C}$$

$$+\tau(\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)).$$

Combine like terms:

$$\mathcal{L}(\tau) = -\frac{1}{2}\tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 - \frac{\tau^2}{4C} + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)).$$

Step 6: Differentiate w.r.t. τ

The Lagrangian is:

$$\mathcal{L}(\tau) = -\frac{1}{2}\tau^2(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 - \frac{\tau^2}{4C} + \tau(\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)).$$

To find the optimal τ, differentiate $\mathcal{L}(\tau)$ with respect to τ:

$$\frac{\partial \mathcal{L}}{\partial \tau} = -\tau(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 - \frac{\tau}{2C}$$
$$+(\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)).$$

Setting the derivative to zero to find the critical point:

$$-\tau(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 - \frac{\tau}{2C} + (\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)) = 0.$$

The differentiated Lagrangian is set to zero:

$$-\tau(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 - \frac{\tau}{2C} + (\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)) = 0.$$

Rearranging the terms to isolate τ:

$$\tau\left[(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + \frac{1}{2C}\right] = (\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)).$$

Solving for τ:

$$\tau = \frac{(\rho \cdot I(y_t = 1) + I(y_t = -1))(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t))}{(\rho \cdot I(y_t = 1) + I(y_t = -1))^2\|\mathbf{x}_t\|^2 + \frac{1}{2C}}.$$